IBM Software Group

An IBM Proof of Technology

Discovering the value of SOA featuring WebSphere Process Integration for your organization

Created for TechWorks

PoT.WebSphere.06.4.031.01

IBM SOA Foundation

WebSphere Process Integration

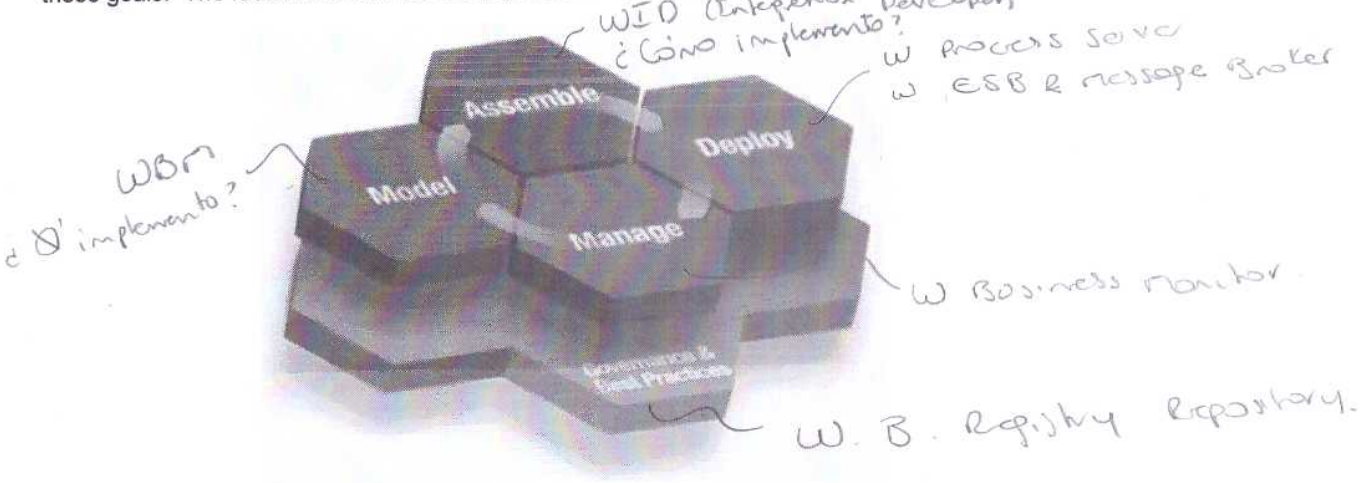
Introduction

In this lab exercise, you will be implementing a business process from the ground up for verifying accounts. This business process will be called **SimpleAccountVerification**, and will be used to process requests to open accounts. This business process will consist of activities such as performing credit checks, risk assessments, approvals, and pricing plan determination.

The main objectives include:

- Business flexibility
- Better business processes
- Easier integration
- Reuse of assets
- Reduction of risk

IBM's SOA Foundation and WebSphere Process Integration software portfolio will be used to achieve these goals. The focus will be on the Model, Assemble, and Deploy phases of the SOA lifecycle.



You will use the **WebSphere Business Modeler** to define the business model, and then **WebSphere Integration Developer** for application assembly, deployment, and testing. You will also use the **WebSphere Service Registry and Repository** to reuse existing standard business object and service definitions.

Once the business process has been implemented and deployed, you will also perform several iterative changes to the process to illustrate how quickly this can be accomplished. This type of business flexibility is one of the primary benefits of SOA.

Important: If you encounter problems during the course of this lab, please call the attention of the class instructor or any of the lab assistants.

Scenario

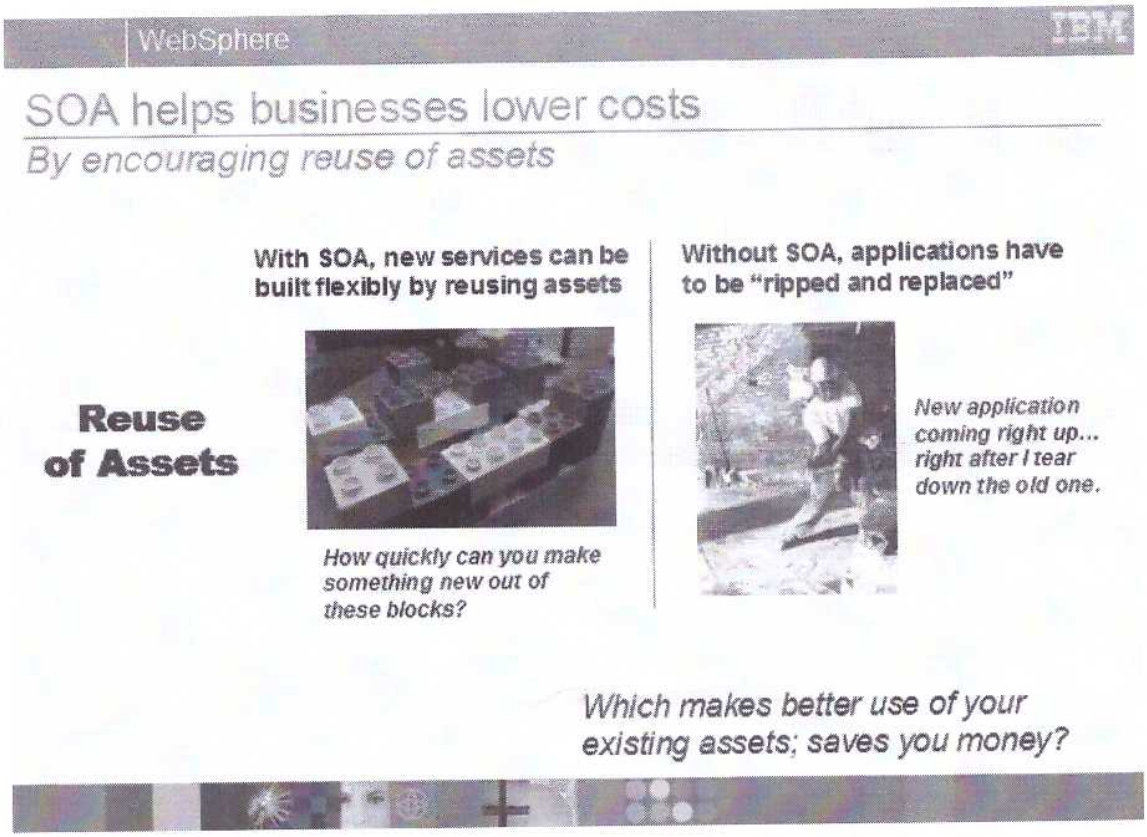
Your organization has several core business processes that need to be significantly improved. These processes are composed of business activities which are currently manual tasks, or implemented using standalone systems. These processes are currently not implemented end-to-end, which significantly limits business flexibility.

In this exercise, you have been assigned to integrate a very simple business process for handling account applications. This is your organization's first SOA project, so we will keep it simple and incremental. This will involve a very rudimentary process of determining whether a request to open an account will be approved or rejected. This generic example can apply to loan applications, revolving credit applications, or other similar scenarios.

Lab Module Overview

- **Part 1: Explore the Service Registry**
- **Part 2: "Model" the Business Process**
- **Part 3: "Assemble" the Business Process**
- **Part 4: "Deploy" the Business Process**
- **Part 5: IT Flexibility**
- **Part 6: Business Rules**
- **Part 7: Human Tasks**
- **Part 8: Connectivity Flexibility with Mediations**
- **Part 9: Dynamic Service Invocation**

Part 1: Explore the Service Registry for Reuse

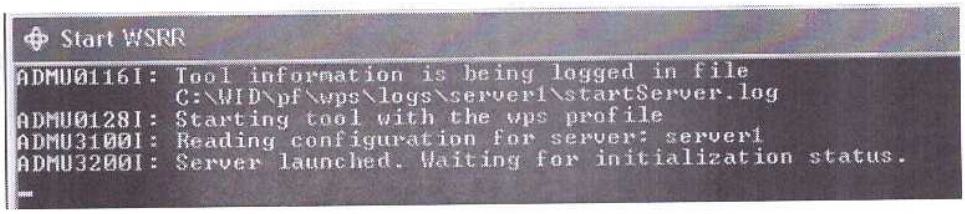


Before you begin the SOA lifecycle, you will first explore the reusable assets that are available and stored in the WebSphere Service Registry and Repository.

____ 1. Start the WSRR Server

___a. Double-click on the **Start WSRR** icon on the desktop.

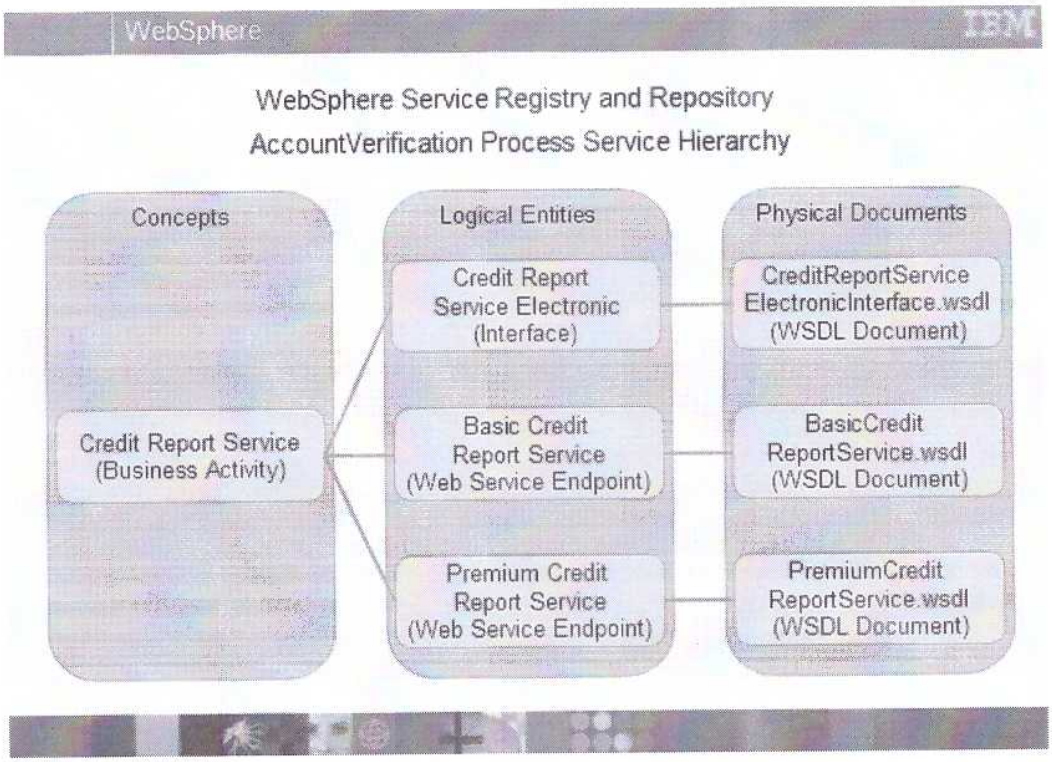
A Command Prompt window will appear.



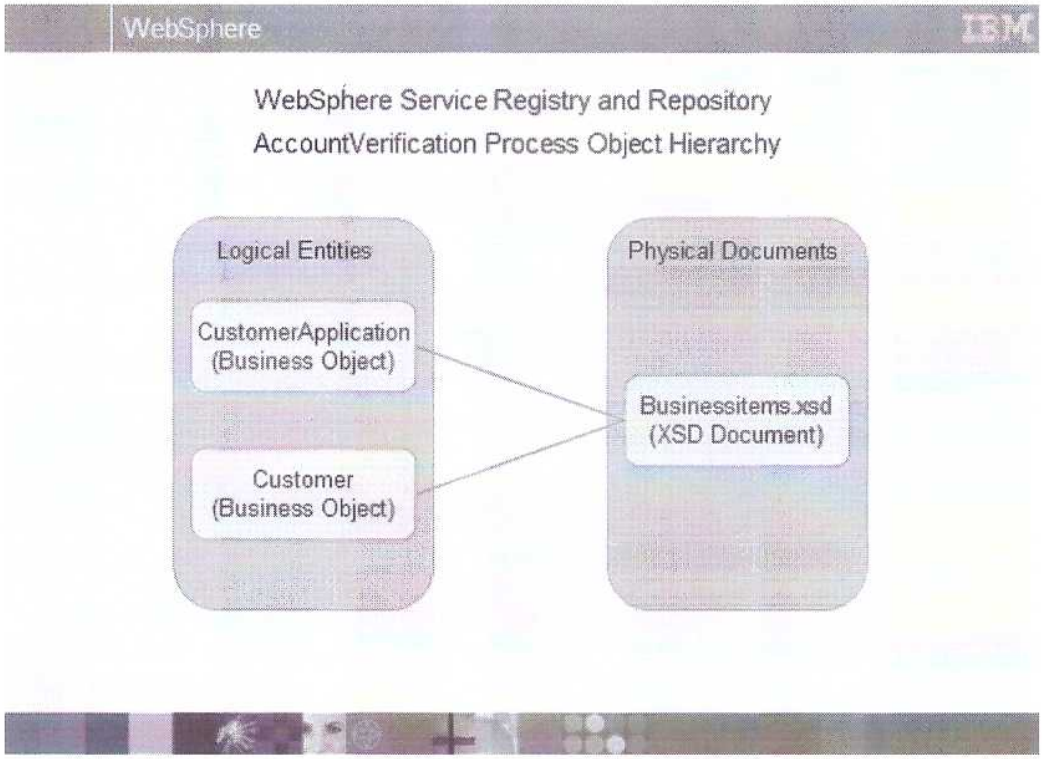
This will start the WebSphere Service Registry and Repository (WSRR) server. The Command Prompt window will automatically close when the server has started.

The next step is to briefly review the documents and concepts that have already been loaded into the WebSphere Service Registry and Repository (WSRR). We will only focus on the artifacts and metadata associated with the services to be used in an Account Verification process. However, feel free to explore the other contents of the registry and repository. WSRR will help promote the reuse of existing services and accelerate the adoption of standards.

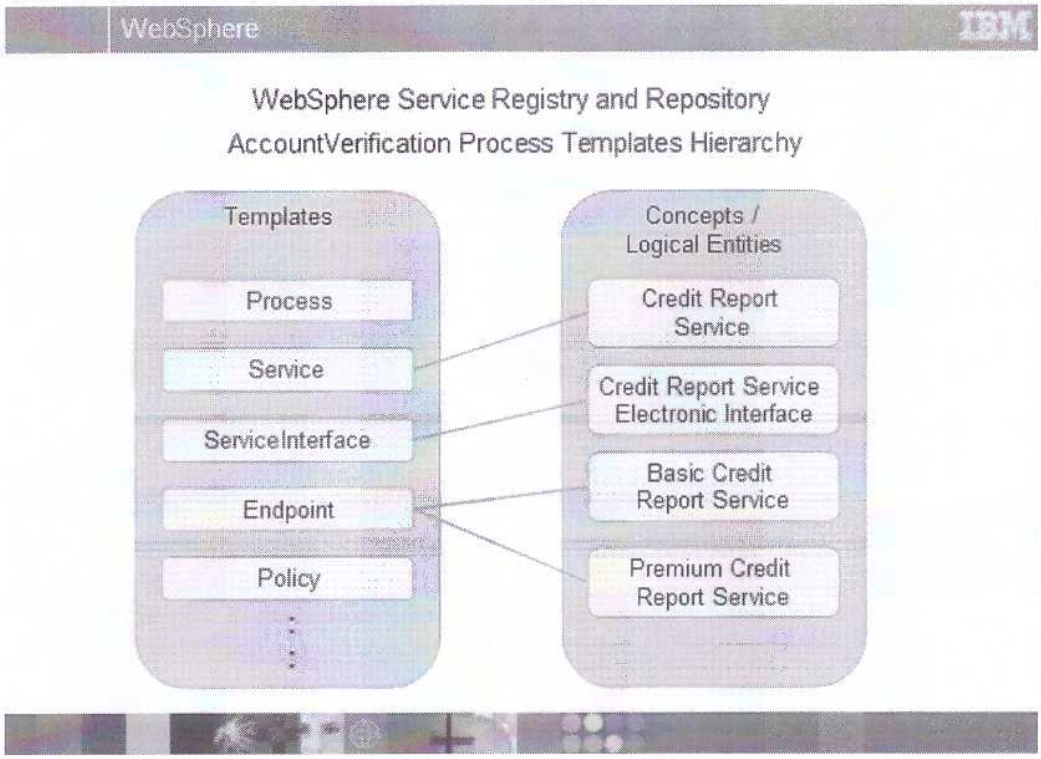
The following diagrams outline the concepts, logical entities, and physical documents which we will explore in the WebSphere Service Registry and Repository.



The Credit Report Service will be used in an Account Verification Business Process that is similar to what you will be implementing in the lab exercises. There are two types of Credit Report Services, Basic and Premium, which are actually web services defined in WSDL documents. WSDL, or the Web Services Description Language is a standard specification for describing web services.



The **CustomerApplication** and **Customer** business objects will be used in both the SimpleAccountVerification process, and the more evolved AccountVerification process. The **Customer** object is actually also an attribute of the **CustomerApplication** object. In other words, the **CustomerApplication** object contains a **Customer** object.

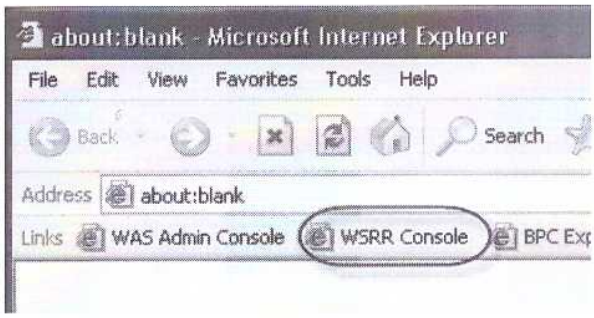


Templates were used to define the concepts and logical entities of the AccountVerification process stored in the WebSphere Service Registry and Repository. For example, the Service template was used to create the Credit Report Service concept.

___b. Double-click on the **Internet Explorer** icon on the desktop. There is also an **Internet Explorer** icon in the **Quick Launch** Bar.

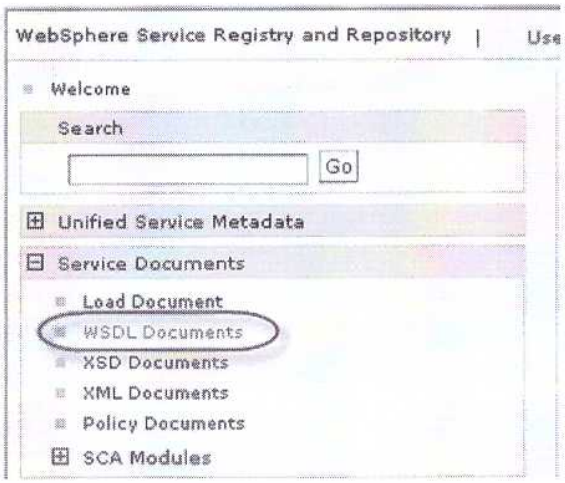
or

___c. From the **Internet Explorer** window, click on the **WSRR Console** link. You can also open the URL http://localhost:9083/ServiceRegistry/ in the web browser.

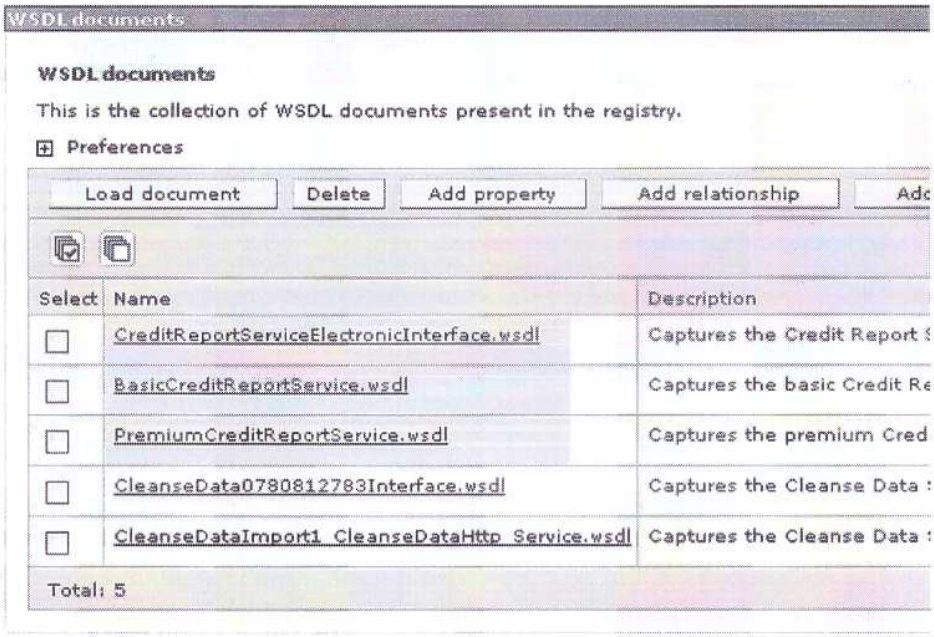


If the WebSphere Service Registry and Repository (WSRR) console page does not appear, then verify that the WSRR Command Prompt window started earlier has disappeared. If not, wait for the Command Prompt window to disappear, and then click on the **WSRR Console** link again.

___d. When the WSRR Console page appears, click on the **Service Documents -> WSDL Documents** link on the navigation bar on the left.



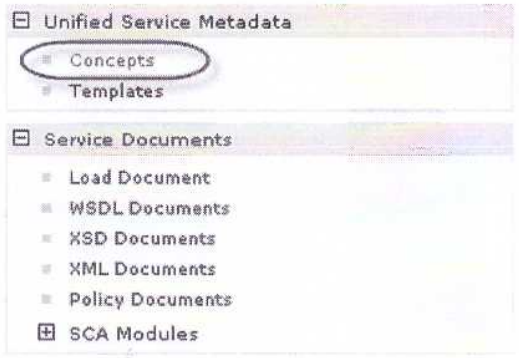
This will display the WSDL files that have already been loaded into the WebSphere Service Registry and Repository (WSRR).



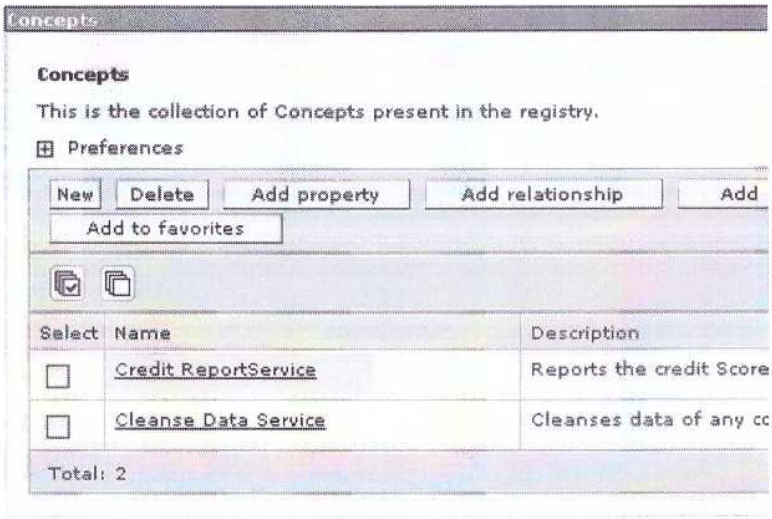
The three WSDL files highlighted above are used in the AccountVerification process. The **BasicCreditReportService.wsdl** document contains the service, port and binding definition of a Basic Credit Report web service which will be invoked by the AccountVerification process. The **PremiumCreditReportService.wsdl** describes a similar credit report web service which will be used for premium customers. The **CreditReportServiceElectronicInterface.wsdl** contains the common interface definition for both BasicCreditReportService and the PremiumCreditReportService.

Feel free to explore further by clicking on the WSDL document links.

___e. Click on the **Unified Service Metadata -> Concepts** link.

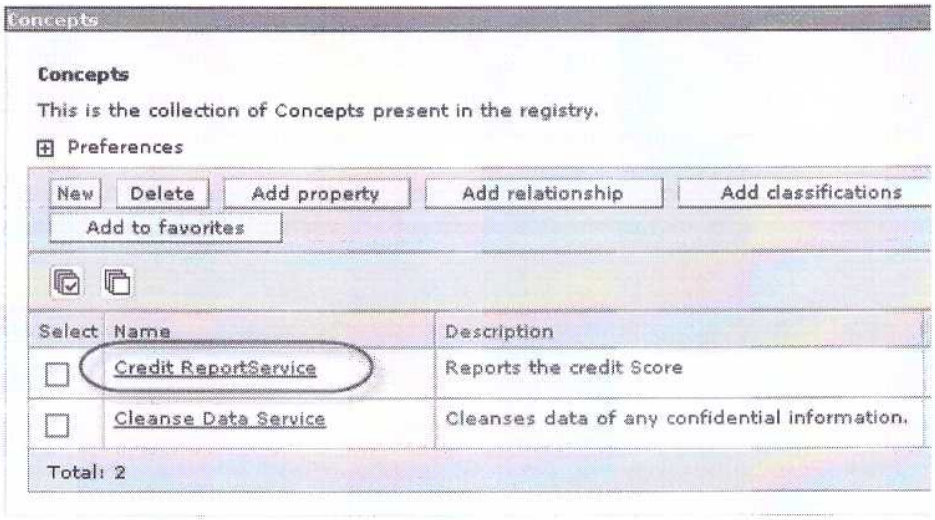


This will display all concepts defined in WSRR. Concepts contain the metadata information associated with any document or artifact. This allows us to describe an abstract business concept, such as a business process or business activity.

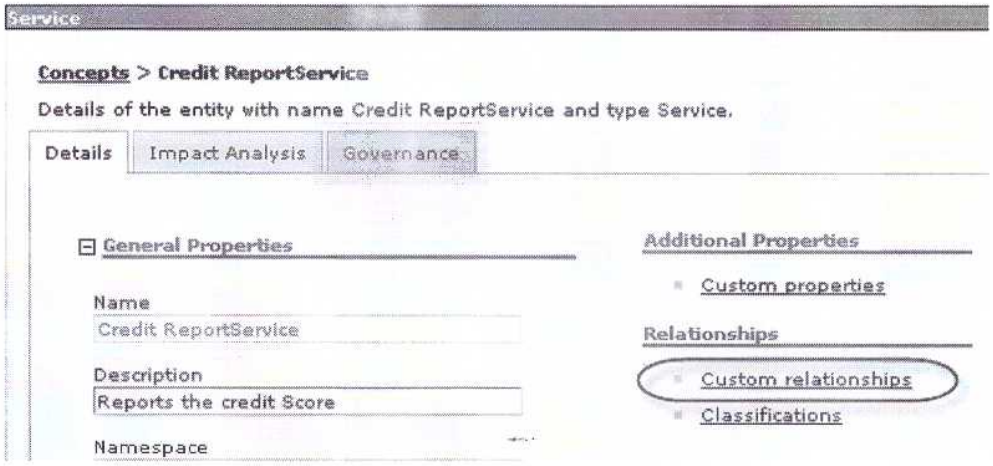


The **Credit ReportService** concept highlighted above represents a business activity which is part of the **AccountVerification** process. At this level, the Credit **ReportService** is only an abstract concept, but relationships have been defined to associate this concept to actual web service endpoints. The endpoints are the BasicCreditReportService and the PremiumCreditReportService shown previously.

___f. Click on the **Credit ReportService** link.

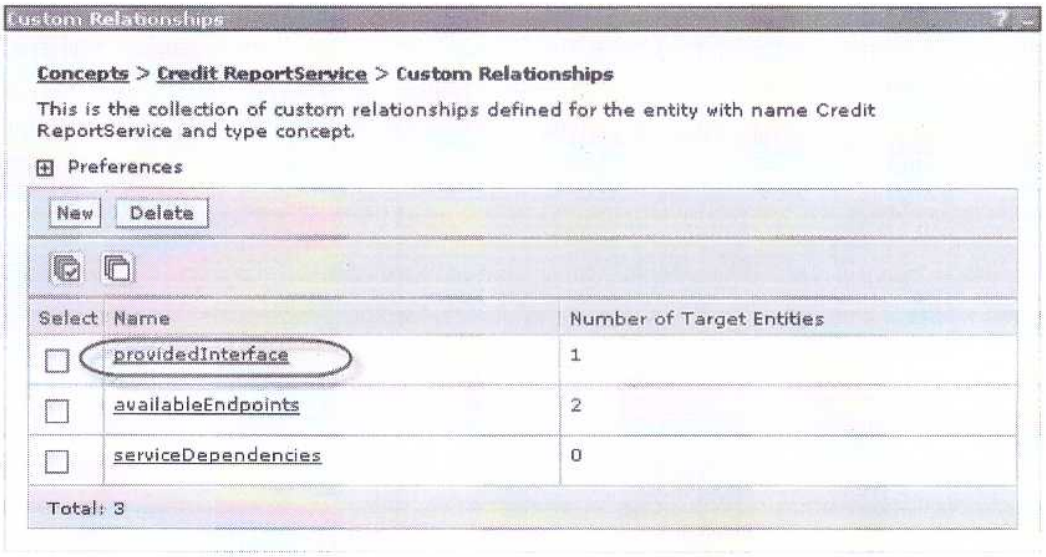


___g. Click on the **Custom relationships** link.

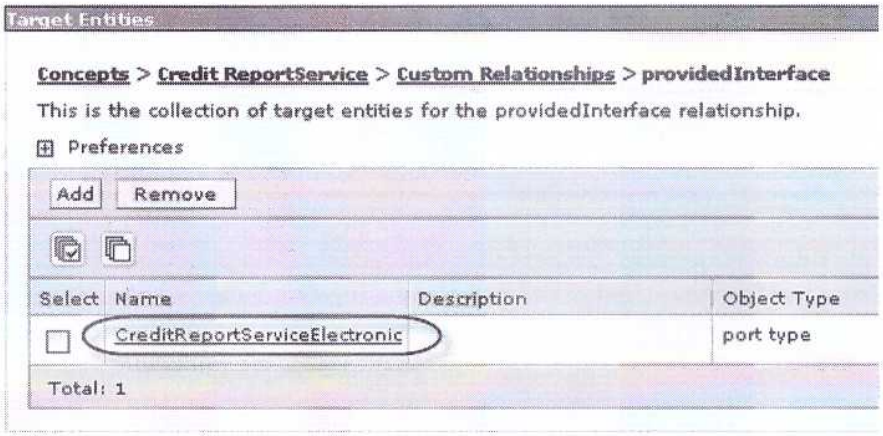


Take note of the **Governance** tab. The WebSphere Service Registry and Repository also provides several governance capabilities which are highly recommended for any SOA project, but we will not apply governance in this exercise for the sake of simplicity. Governance is covered in more depth in other available Proof-of-Technology sessions, demos, Redbooks, etc.

___h. Click on the **providedInterface** link.



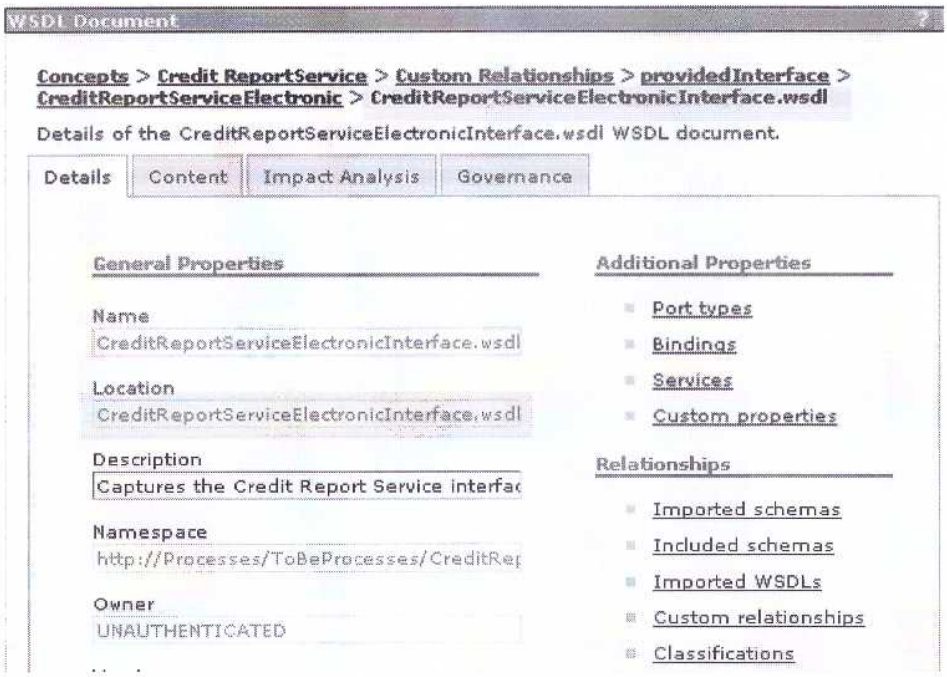
___i. Click on the **CreditReportServiceElectronic** link.



CreditReportServicelElectronic is a port type which represents the set of operations provided by the two Credit Report web services shown earlier, BasicCreditReportService and PremiumCreditReportService.

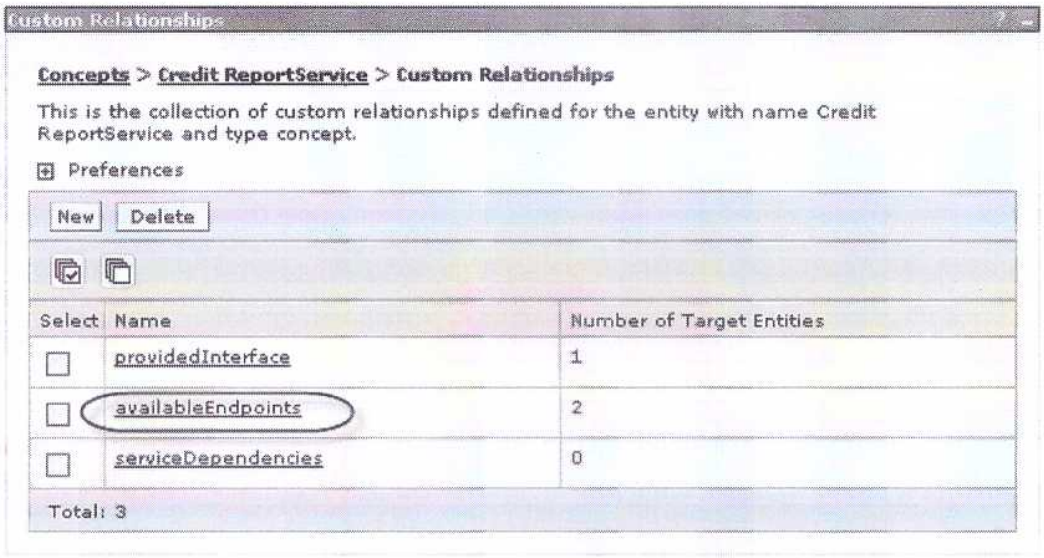
___j. Click on the **Source document** link.

This will display where the relationship between the Credit Report Service interface and its WSDL document (CreditReportServiceElectronicInterface.wsdl) is defined.



___k. Click on the **Custom Relationships** link.

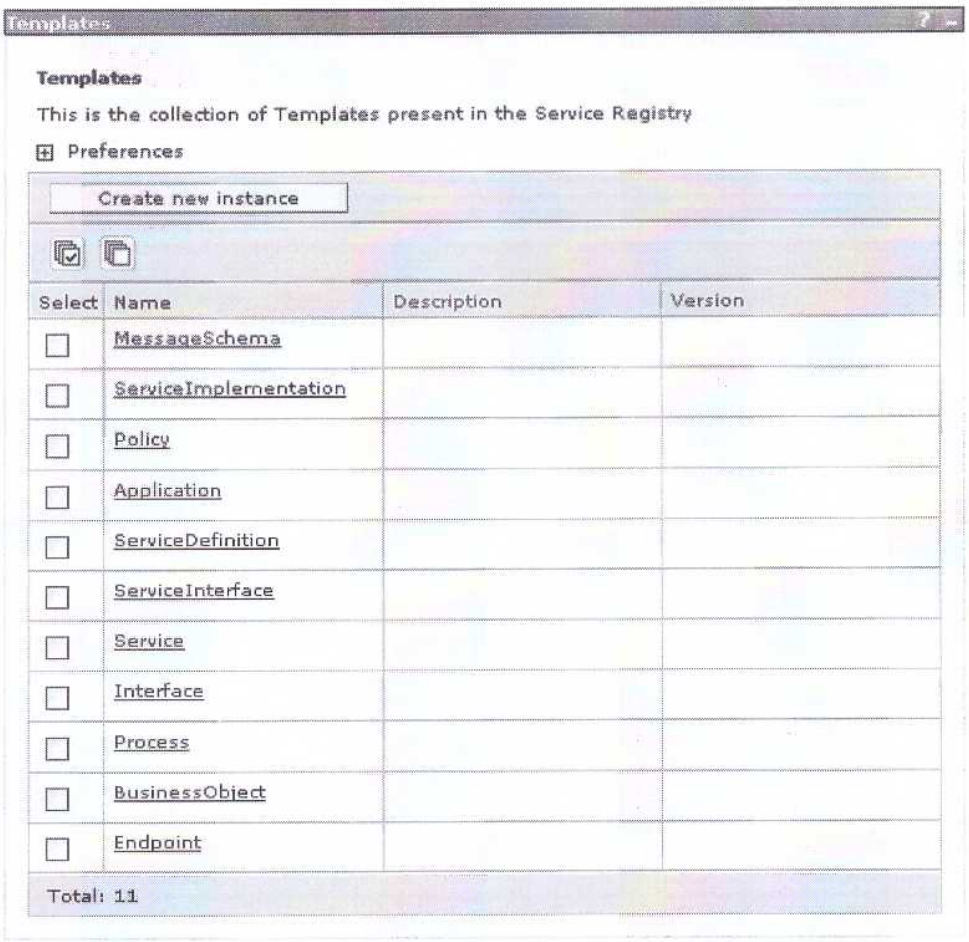
___l. Click on the **availableEndpoints** link.



This will display the two web service endpoints associated with the Credit Report Service.

___**m.** From the navigation bar, click on the **Templates** link.

This will display all the templates currently stored in the Service Registry.

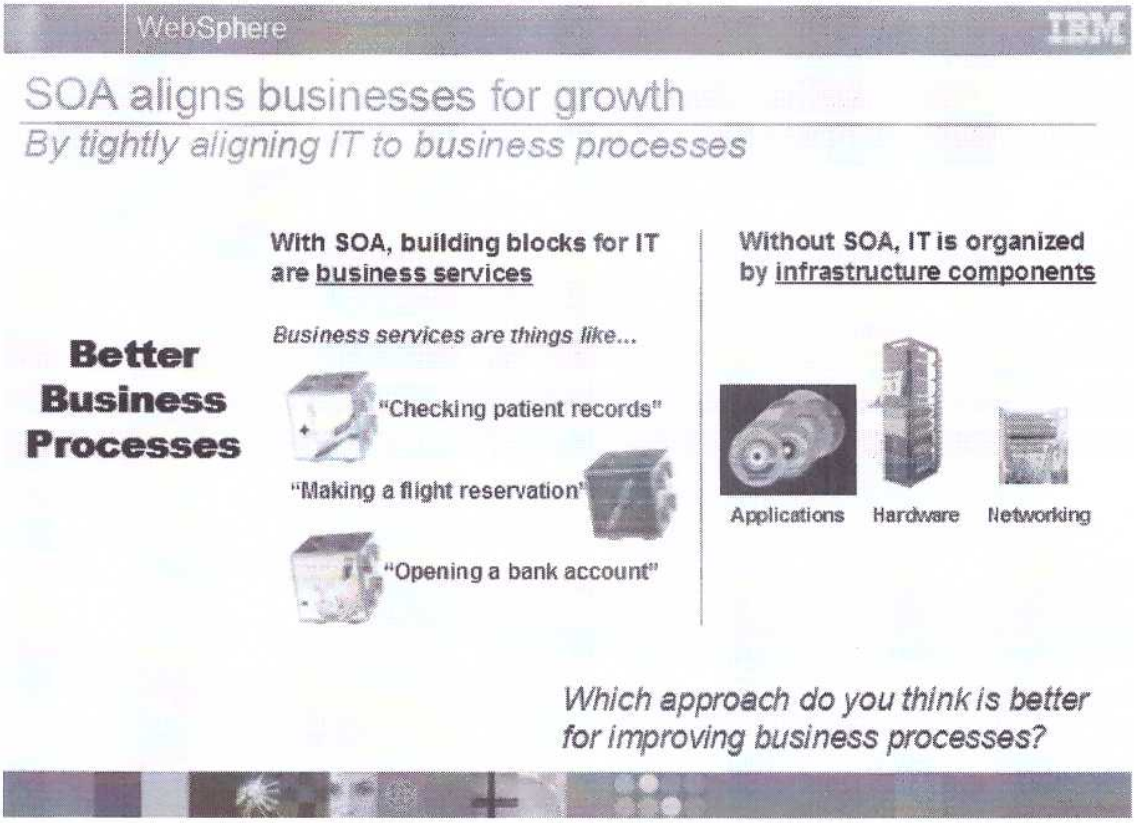


Feel free to continue browsing the Service Registry, but do not make any changes to the existing entities because this might affect the successful completion of the lab exercises.

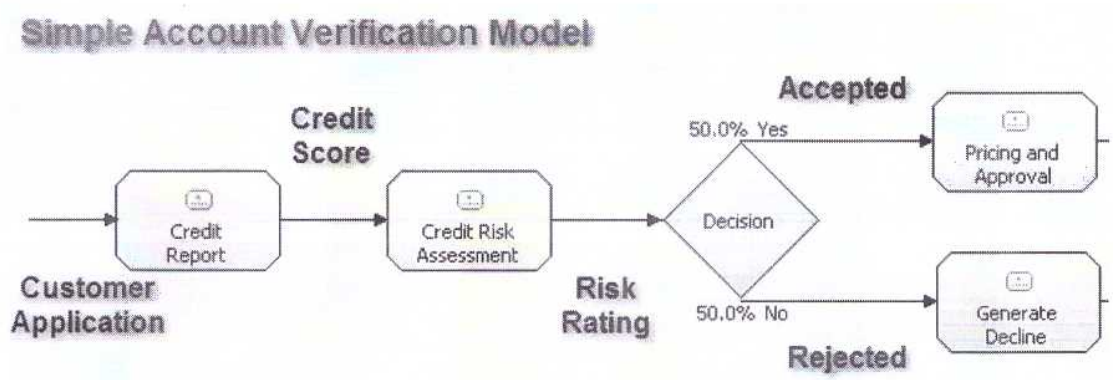
___n. Close the **Internet Explorer**.

Please wait for the next lecture before proceeding with the lab.

Part 2: "Model" the Business Process



In this **Model** phase, you will assume the role of a **Business Analyst**. You have been given the responsibility of documenting an existing business process for account verifications. This is a simple process which involves customers submitting a request to open an account. The customer submits the application form with the required information, and then a credit report is performed to determine the applicant's credit score. A credit risk assessment is then completed based on the credit score, which will result in a risk rating of either HIGH or LOW. A request with a high risk rating will automatically be rejected. A request with a low risk rating will be approved and a pricing plan will be determined.



____ 1. Start the WebSphere Business Modeler

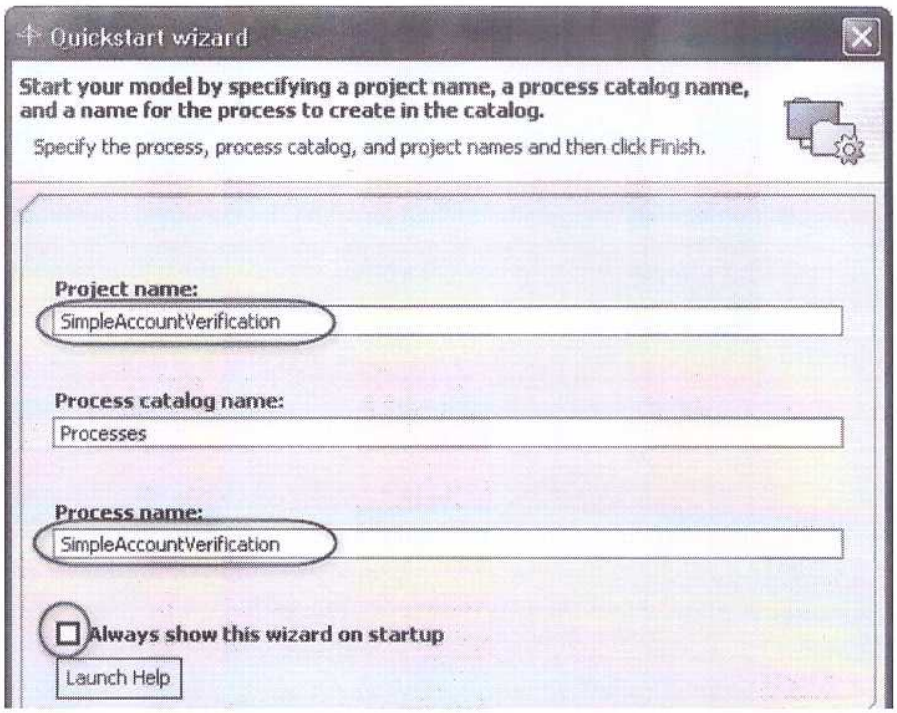
___a. Double-click on the **Model - SimpleAccountVerification Model** icon on the desktop.

The WebSphere Business Modeler will appear.

___b. On the **Quickstart Wizard**, specify the following. (Refer to screenshot)

Project Name	SimpleAccountVerification
Process Name	SimpleAccountVerification

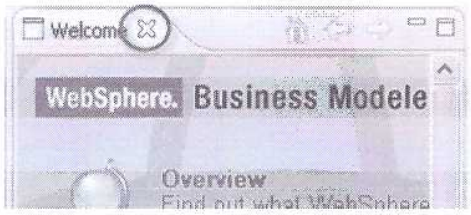
Use the default for the **Process catalog name** field, and uncheck the **Always show this wizard on startup** option.



___c. Click on **Finish**.

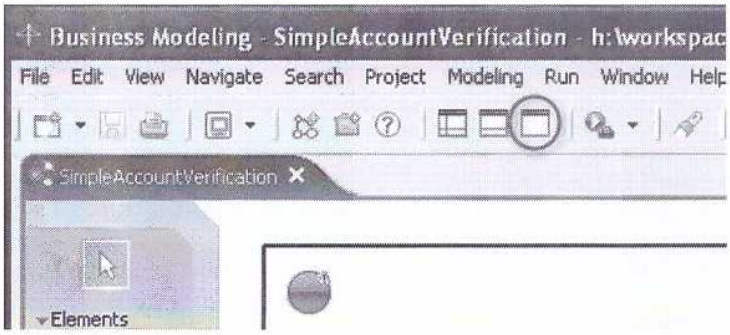
___d. Maximize the **WebSphere Business Modeler** window if needed.

___e. Close the **Welcome** view.



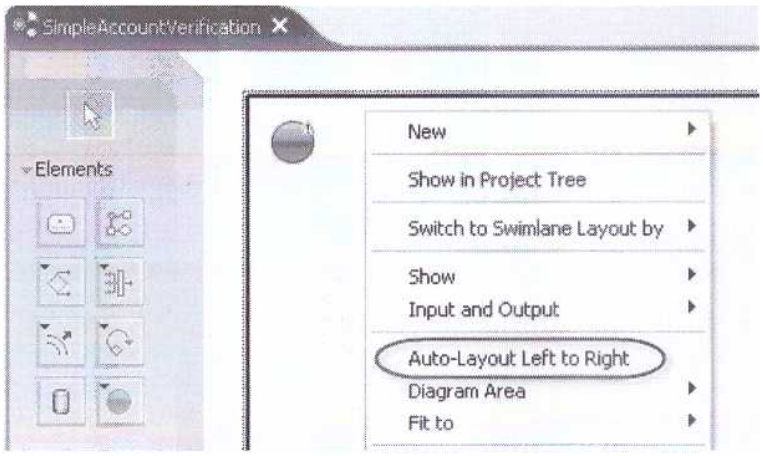
____ 2. Add tasks to the Business Model

___a. From the toolbar, click on the **Apply 1-pane layout** button. (Refer to screenshot)



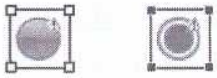
This will simplify the layout and hide the advanced features and views for now.

___b. In the canvas or white-space area, **right-click** and then select **Auto-Layout Left to Right**.



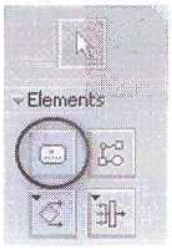
This will compress all the default elements and make them easier to find. By default, the business model will initially have **Process start** and **Process stop** elements positioned in the corners of the canvas, but should now be visible.

___c. Select the two default process elements and press the **Delete** key. You will not need these elements for the business model you are creating.



Tip: You can undo any incorrect actions by either pressing **Ctrl-z**, or selecting **Edit->Undo** from the menu bar.

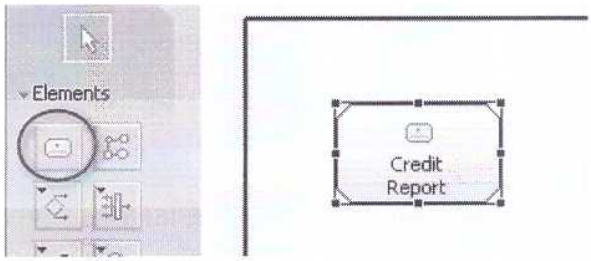
___d. From the palette on the left, select the element **Create local task**. (Refer to the screenshot.)



This element will represent a business task in the business process you are diagramming. Keep in mind that this element typically should not represent programs, applications, databases, or any other software component, but rather high-level activities composing a business process.

___e. Click on the canvas to drop the local task element. By default, the name of the element should be in edit mode. Label the element as **Credit Report**.

Hint: You do not need to drag and drop.

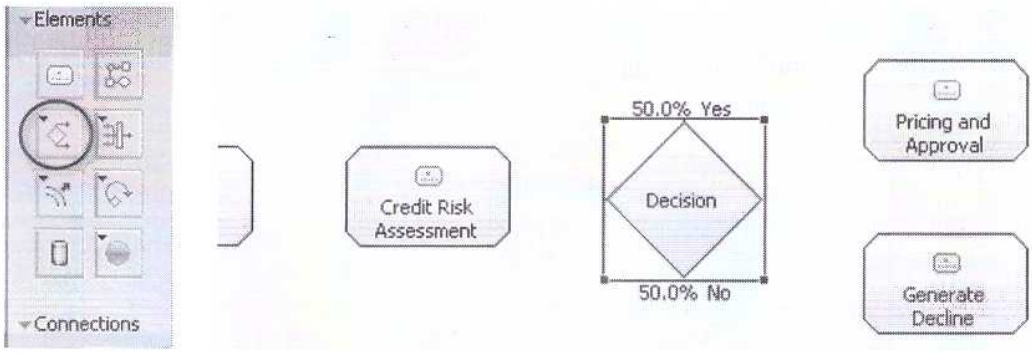


To rename the element, just highlight the element, wait 1-2 seconds, then click on the element again. The label will be in edit mode.

___f. Select three more local task elements and drop them into the canvas. Rename these new elements as **Credit Risk Assessment, Pricing and Approval** and **Generate Decline**. Position the elements similar to the screenshot.

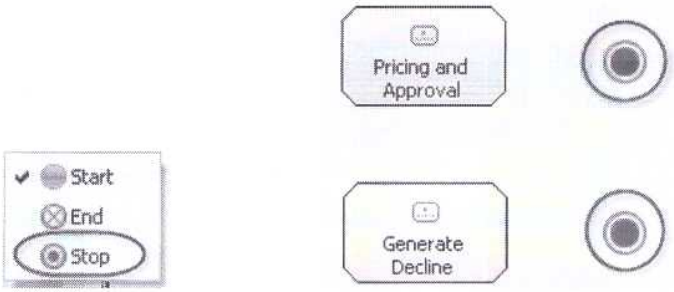
These four elements will represent the four tasks in our simple account verification scenario. These tasks will later be implemented as invocations to applications or external systems.

___g. From the palette, select the element **Create simple decision** and drop it in the middle of the three new elements.



___h. From the palette, click on the small arrow on the element **Create start** to display a submenu of elements.

___i. Select the element **Stop** and drop to the right of both the **Pricing and Approval** element and the **Generate Decline** element.



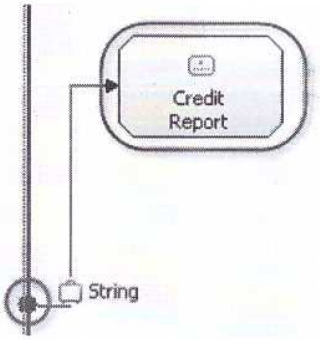
___j. Click on **Ctrl-s** to save your current progress.

____ 3. Connect tasks to define process flow

___a. From the palette, click the **Connections** button. 

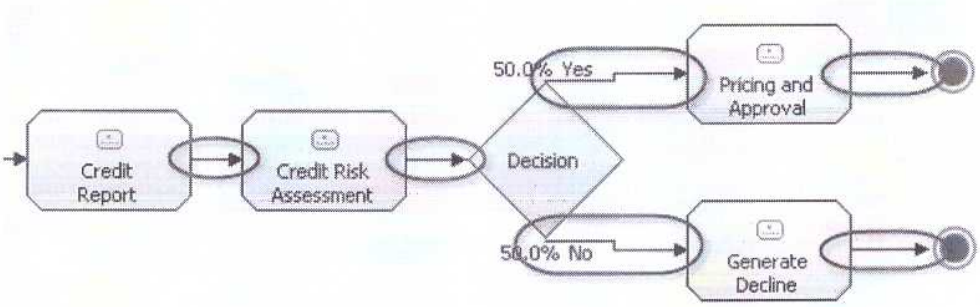
This will allow you to start defining the flow through the different tasks to represent the business process flow.

___b. Click on the left border of the canvas, and then click on the **Credit Report** task.



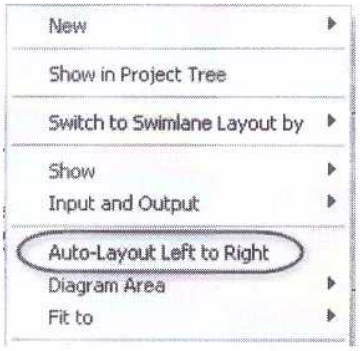
This connection indicates that the **Credit Report** task will be invoked first and will be passed some **String** or text information as a parameter.

___**c.** Complete the following connections:



___**d.** From the palette, click on the **Select** button to switch out of connection mode. 

___**e.** Right-click on an empty space in the canvas. From the popup menu, select **Auto-Layout Left to Right**.



The wiring of all the business tasks to define the process flow is now complete. You will now configure the business item that will be used by the business tasks.

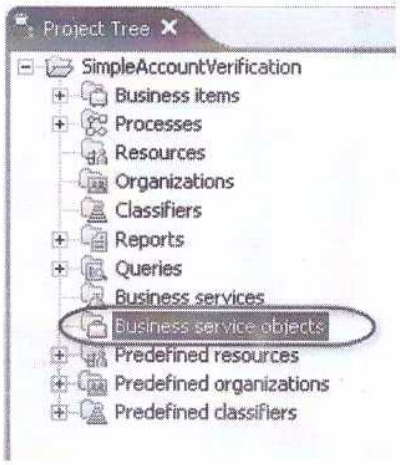
___**f.** Click on **Ctrl-s** to save your current progress.

___g. Close the **SimpleAccountVerification** Model editor.

___h. From the toolbar, click on the **Apply 4-pane layout** button

This layout will expose more views and information about the business model.

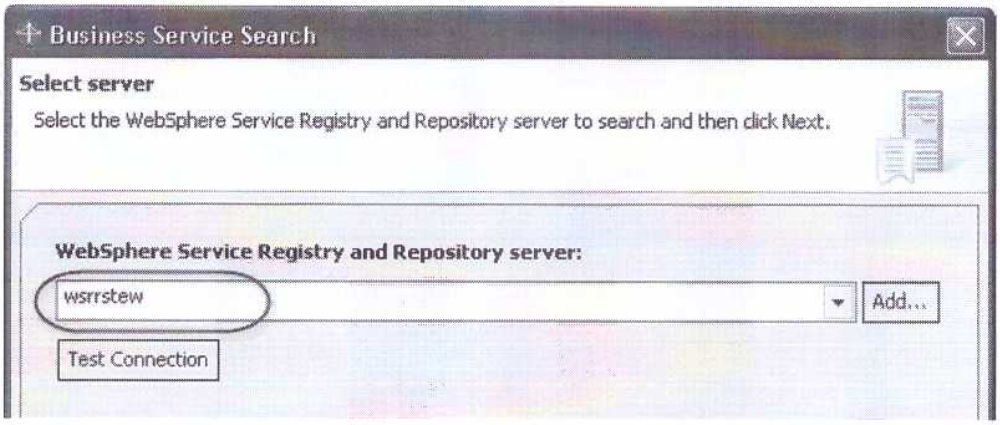
___i. From the **Project Tree** view, expand the **Project** list and select **SimpleAccountVerification -> Business service objects**.



The next step is to reuse existing business object definitions to promote standards. In the **SimpleAccountVerification** business process, the customer information required by the verification process will be stored in a standard business object called **CustomerApplication**. The CustomerApplication object definition will be retrieved from the WebSphere Service Registry and Repository server and stored in the **Business service objects** folder highlighted above.

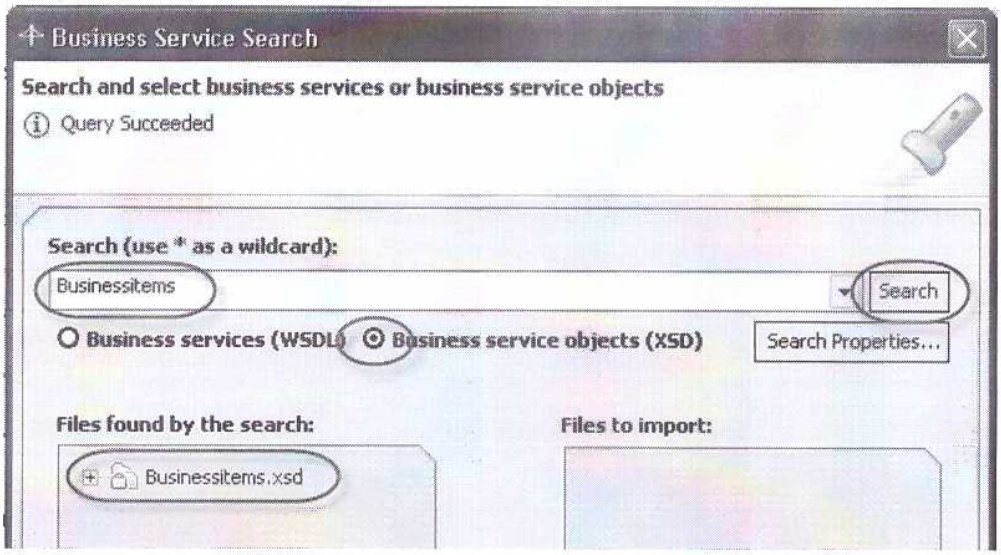
___j. From the **Project Tree**, right-click on the **Business service objects** folder. From the popup menu, select **Business Service Search**.

___k. Ensure that **wsrrstew** is selected as the **WebSphere Service Registry and Repository server**, then click **Next**.



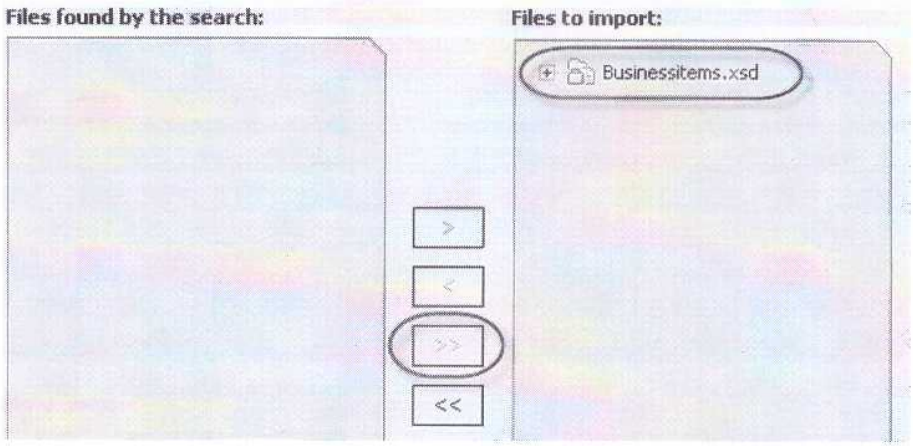
The selection **wsrrstew** is an alias for the WebSphere Service Registry and Repository server you had previously started. The Command Prompt window which appeared when you started the server should have disappeared by now to indicate that the server is now running. If not, wait for the window to disappear before proceeding to the next step.

___l. Specify **Businessitems** as the **Search** criteria (case-sensitive). Select the **Business service objects** option, and then click on **Search**. This will return the file **Businessitems.xsd**.



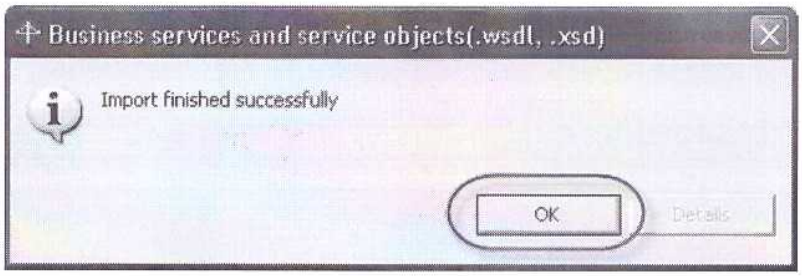
Businessitems.xsd is a XML Schema Definition (XSD) file containing the definition of the CustomerApplication business object. XSD is a standard specification for defining data structures using XML.

___m. Select the All button [>>] to move Businessitems.xsd to the **Files to import** column. Click on **Finish**.

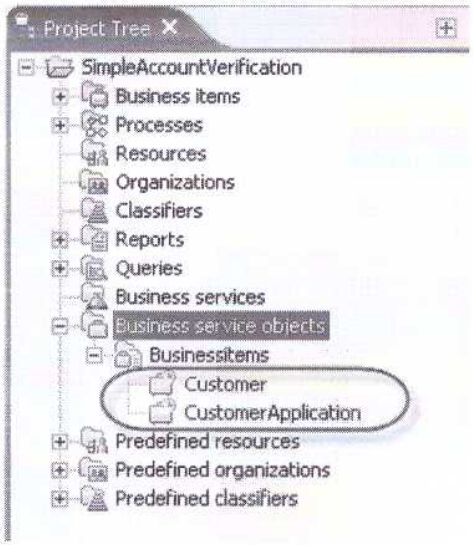


This will import the **CustomerApplication** object into the Business services objects folder in the Project Tree view. It will also import another object called **Customer** because it is part of the **CustomerApplication** object.

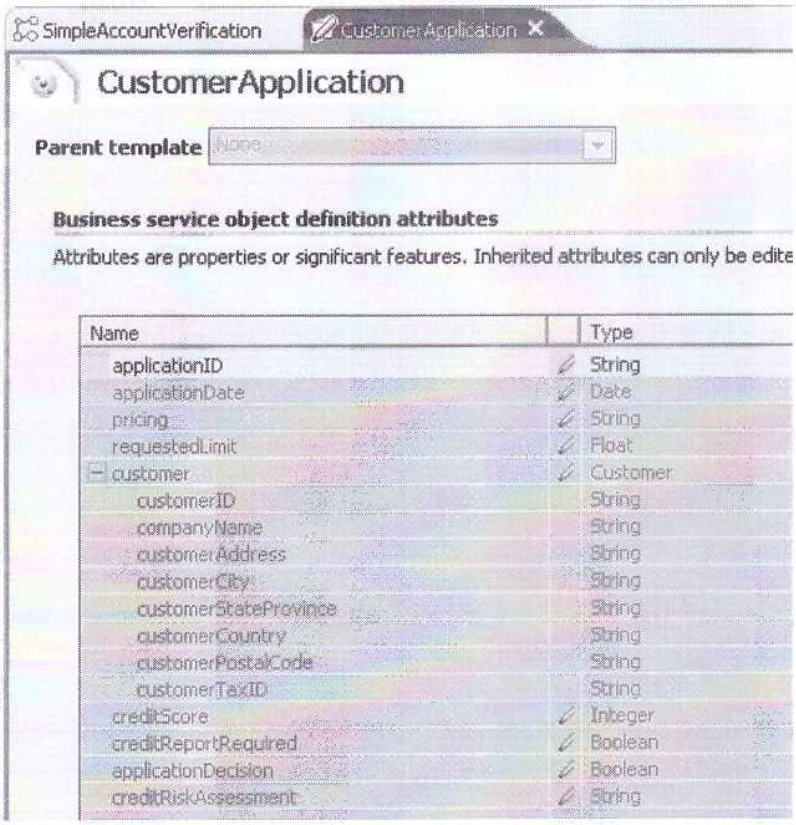
___n. From the Import status information window, click on **OK**.



___o. From the **Project Tree** view, expand the **Business service objects** folder. Ensure that both the **Customer** and **CustomerApplication** business objects exist.



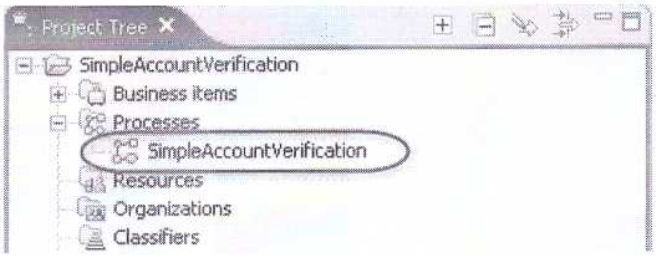
___p. From the **Project Tree** view, double-click on **CustomerApplication**. This will open the **CustomerApplication** business object view.



This is the data structure of the **CustomerApplication** business object. As you can see, it also contains a nested **Customer** business object.

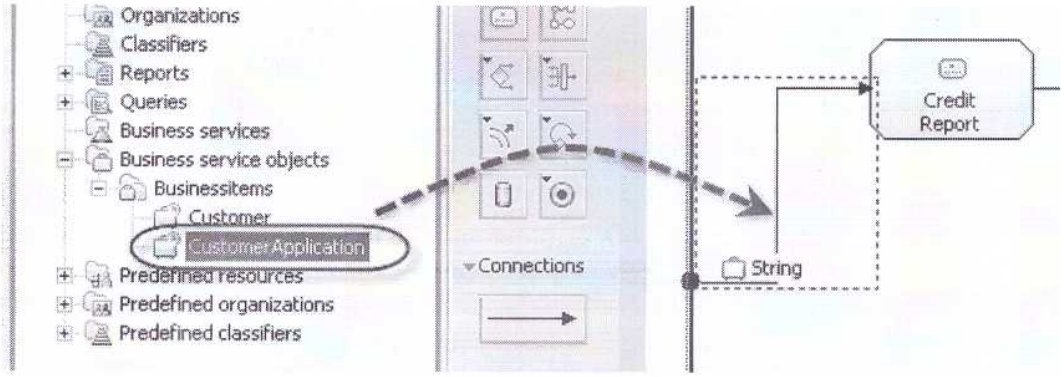
___q. Close the **CustomerApplication** business object view.

___r. From the **Project Tree** view, double-click on **SimpleAccountVerification -> Processes -> SimpleAccountVerification** to open the model editor.

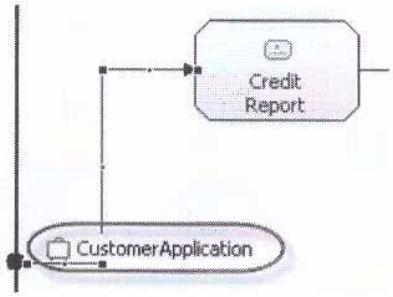


___s. Drag and drop the business item **CustomerApplication** on the connection between the left border of the canvas and the element **Credit Report**.

Hint: A dotted box appears when the object is ready to be dropped on the connection.

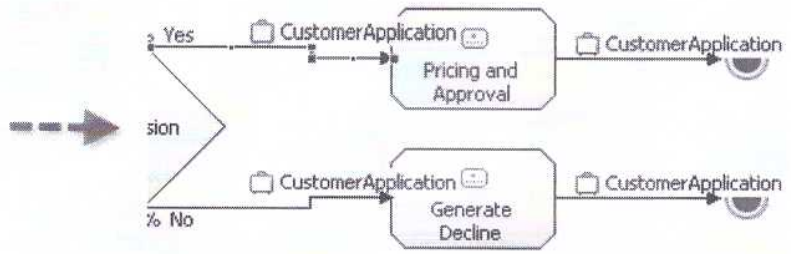


This drag and drop action sets up the business item **CustomerApplication** as the input to the activity **Credit Report**. The connection should now look similar to this:

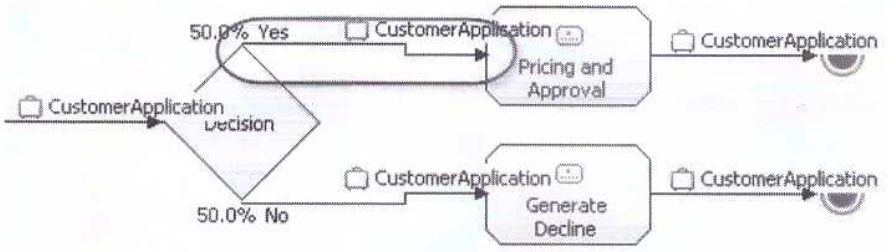


___t. Drag and drop the **CustomerApplication** business item on the rest of the connections.

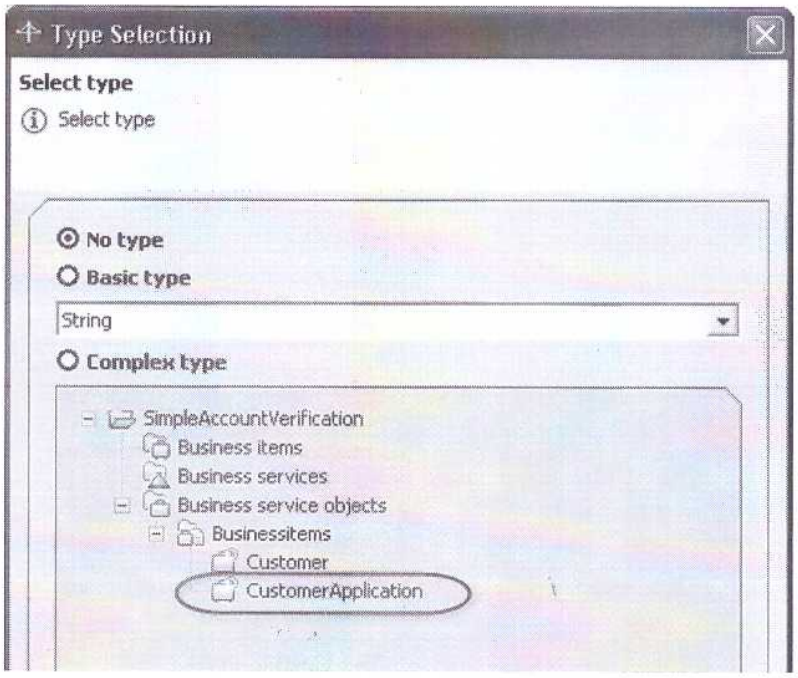
Screenshot continued...



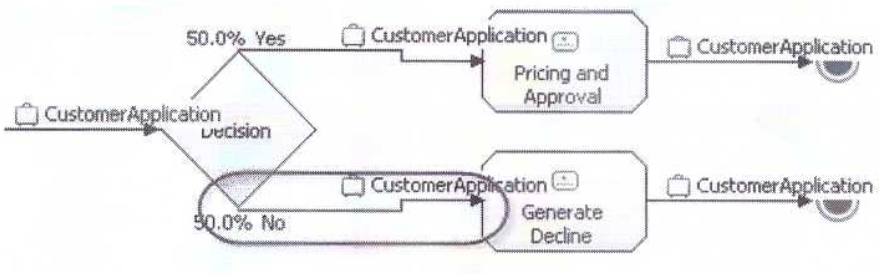
___u. Right-click on the **Yes** connection between the **Decision** activity and the **Pricing and Approval** activity.



___v. From the popup menu, select **Associate Data**. From the **Type Selection** window, select **CustomerApplication**, then click on **OK**.



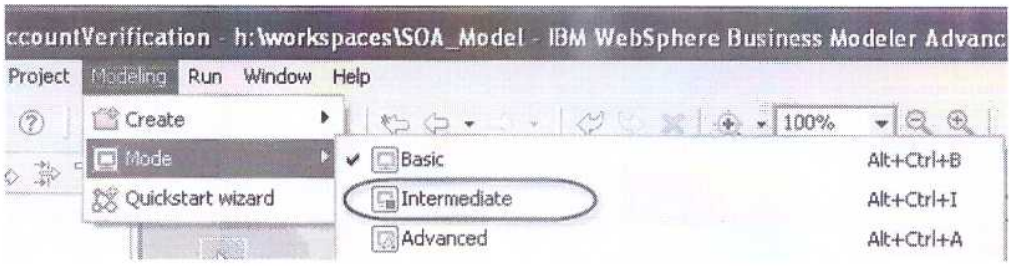
___w. Right-click on the **No** connection. From the popup menu, select Associate Data.



___x. Select **CustomerApplication**, then click on **OK**.

___y. Click on **Ctrl-s** to save your current progress.

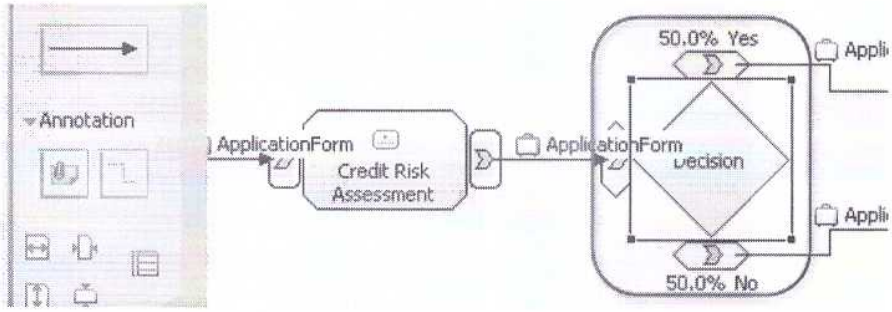
___z. From the menu bar, select **Modeling->Mode->Intermediate**.



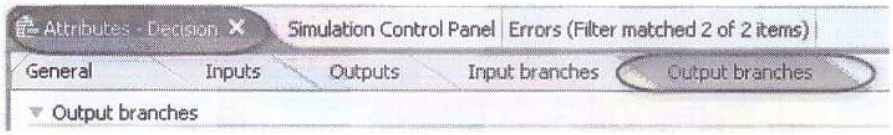
This mode will provide more advanced tooling capabilities. Specifically, this will allow us to provide more sophisticated logic to the **Decision** element. This will also now display more information on the business model diagram.

____ 4. Define the branch logic for the Decision element

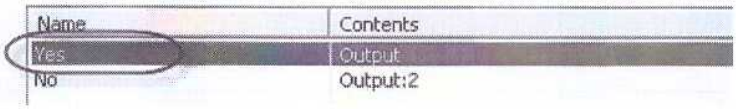
___a. Select the **Decision** element.



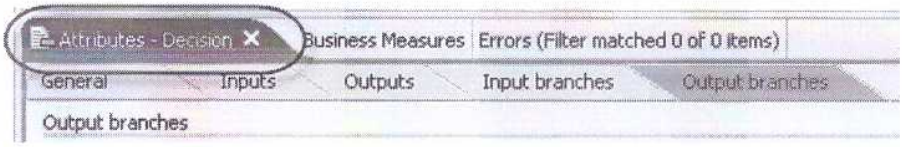
___b. On the **Properties** area at the bottom section of the screen, click on the **Attributes - Decision** tab, and then the **Output branches** tab.



___c. Select the row for the **Yes** output branch.

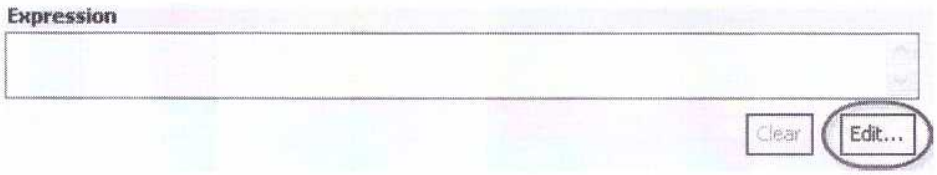


___d. Double-click on the **Attributes - Decision** view titlebar to maximize the view.



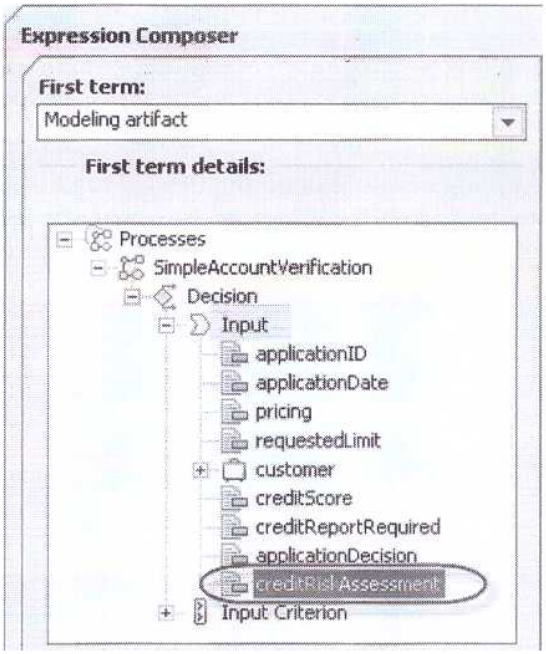
Hint: You can maximize any view by double-clicking on its titlebar. Double-clicking again on the titlebar will restore the view to its original size.

___e. On the lower-right corner of the view, click on **Edit**. Scroll down if necessary.



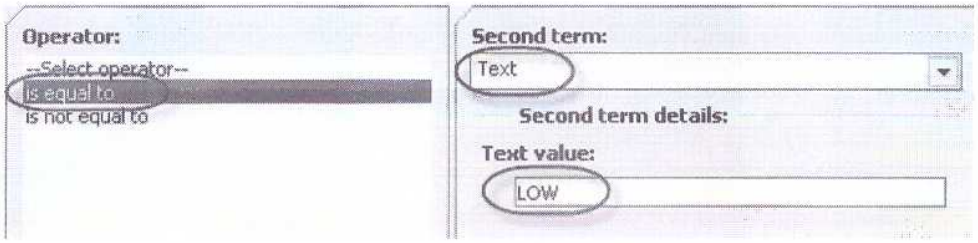
The Expression Builder window will appear.

___f. In the **Expression Composer** section, expand the **First term details** tree list, and then select **creditRiskAssessment**. Ensure that you are selecting **creditRiskAssessment** from the **Input** parent branch, and not the **Input Criterion** branch.



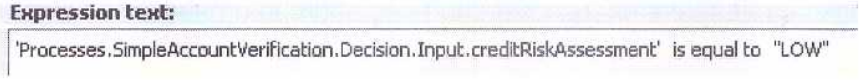
___g. Make the following selections:

Operator	is equal to
Second term	Text
Text value	LOW



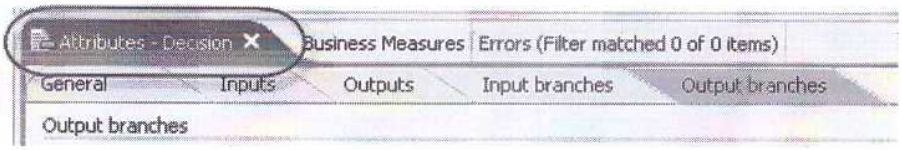
___h. Click on **Apply.** 

___i. Ensure that the **Expression text** contains the following expression, then click on **OK**.



This will also automatically create an inverse expression for the **No** output branch.

___j. Double-click on the **Attributes - Decision** view titlebar to restore the view to its original size.



___k. Press **Ctrl-s** or click on the **Save** button on the toolbar. 

___l. Click on the **Errors** tab and ensure that there are no errors listed.

The business model is now complete.

Important Note: At this point, for the sake of simplicity, we will just use the default names for the operations of the tasks, as well as the names of the input and output variable. Most of the operation names will default to **InputCriterion**. Also by default, input variables will typically be named **input1**, and output variables will be named **output1**.

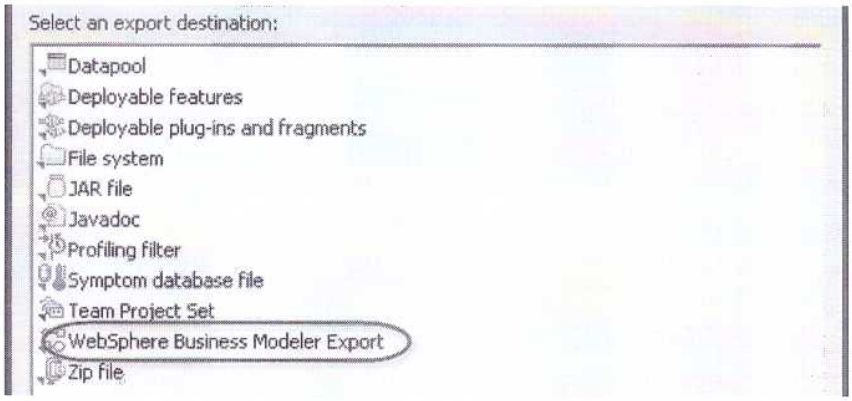
____ 5. Export the business model

The business model will now be exported to the file system as a BPEL version of the model. In the next part of this lab, this BPEL model will be imported into the WebSphere Integration Developer and refined.

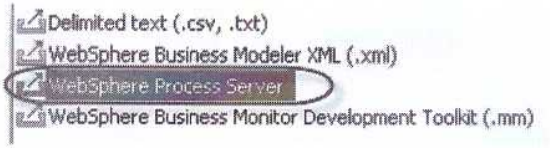
Hint: If the business model was not completed successfully, or was not exported successfully, then you can still proceed to the next part of this lab. A solution version of the model is available if needed.

___a. From the menu bar, select **File -> Export**.

___b. From the **Export** window, select **WebSphere Business Modeler Export**, then click on **Next**.

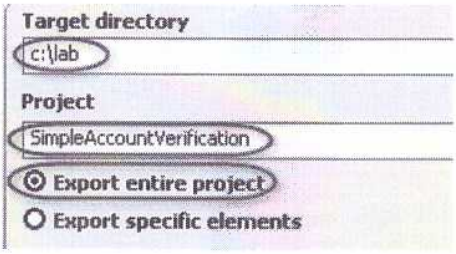


___c. From the next **WebSphere Business Modeler Export** window, select **WebSphere Process Server**, then click on **Next**.



___d. From the next **WebSphere Business Modeler Export** window, specify the following:

Target directory	c:\lab
Project	SimpleAccountVerification
Project selection scope	Export entire project



___e. Click on **Finish**.

The business model has now been exported into the file system and is ready for import into the WebSphere Integration Developer.

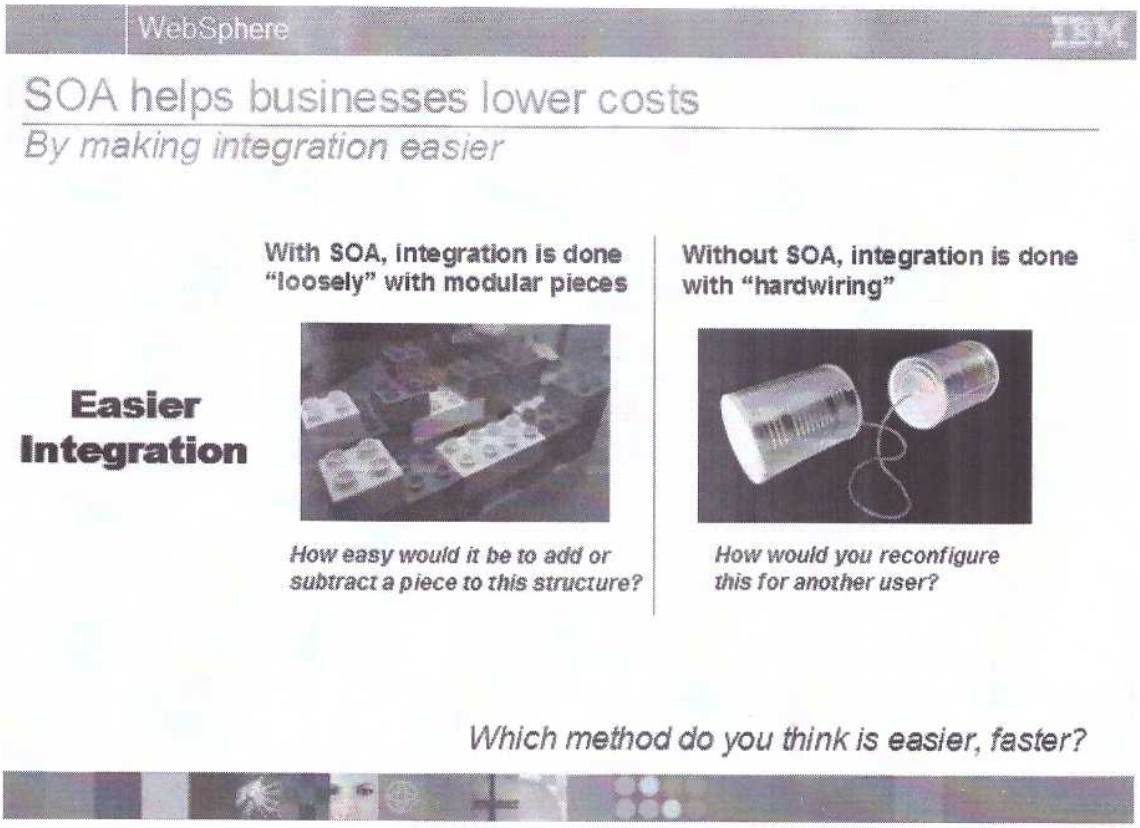
___f. From the **Export finished** window, click on **OK**.

___g. From the main menu, select **File -> Exit** to close the **WebSphere Business Modeler**.

The "**Model**" phase of our SOA approach to implementing this simple business process is now complete.

Please wait for the next lecture before proceeding with the lab.

Part 3: "Assemble" the Business Process



In the **Assemble** and **Deploy** phases, you will now switch roles. You will now be an **Integration Developer** who is responsible for assembling the different service components to compose the business process. The next steps will illustrate how the loose-coupling and building-block approach can be applied to integration development. Through loose-coupling, the different applications and systems required by the business process can be modularized and treated as building blocks. This will make integration at the larger, global, and enterprise-wide scale significantly easier.

____ 1. Import the business model into the WebSphere Integration Developer

___a. Double-click on the **WID - SimpleAccountVerification** icon on the desktop.

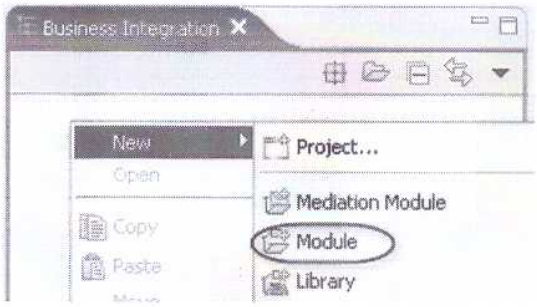
The **WebSphere Integration Developer** Welcome view appears by default.

___b. Maximize the **WebSphere Integration Developer** window if needed.

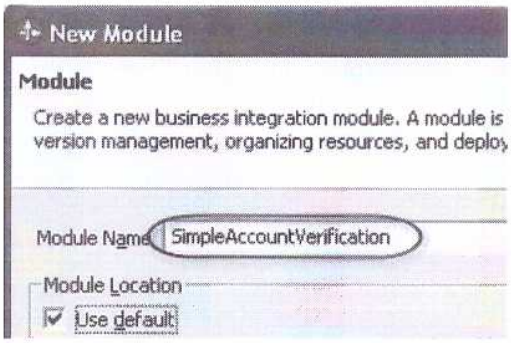
___c. Close the **Welcome** view or click on the **Workbench** icon on the Welcome view.

The Business Integration perspective now appears.

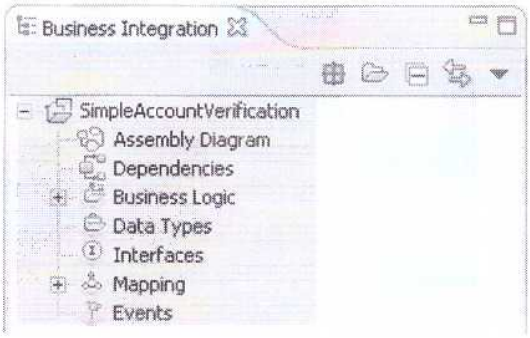
___d. Right-click inside the **Business Integration** view. From the popup menu, select **New->Module**.



___e. In the **New Module** window, specify **SimpleAccountVerification** for the **Module Name**. Accept the other defaults and click on **Finish**.

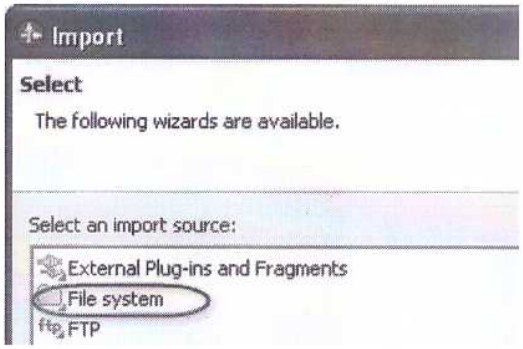


The new module will now appear in the Business Integration view.

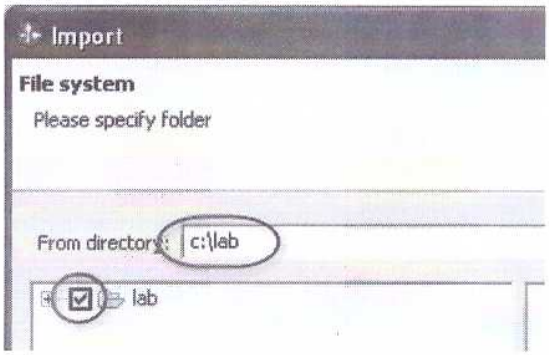


___f. From the menu bar, select **File->Import**.

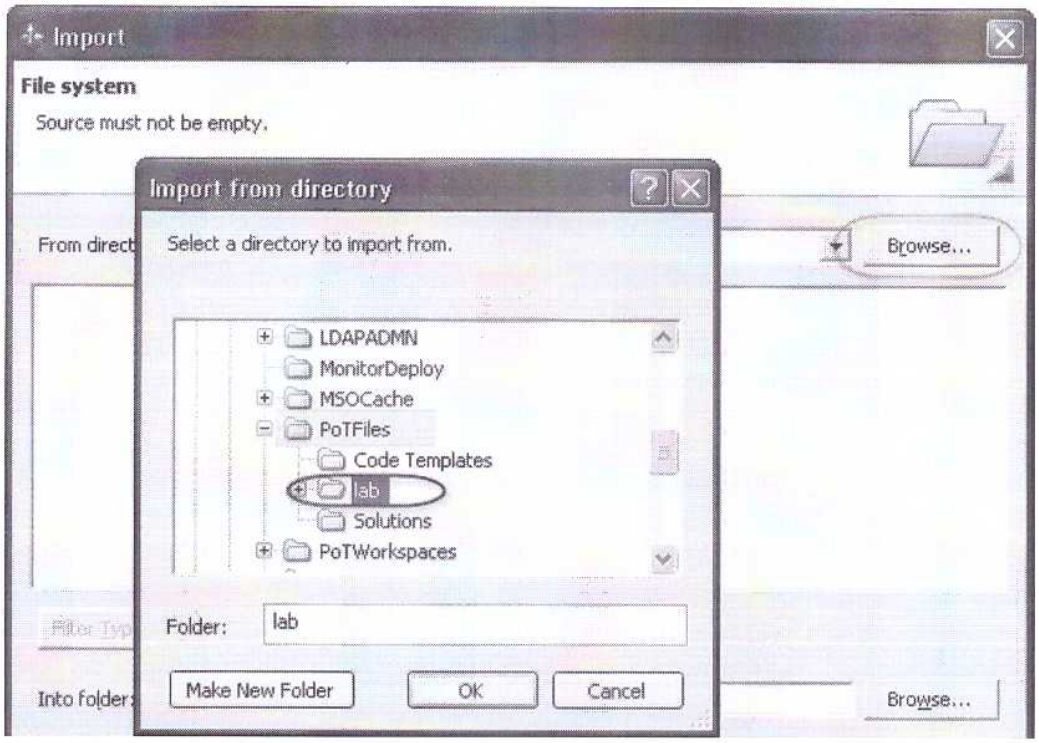
___g. From the **Import** window, select **File System**, then click on **Next**.



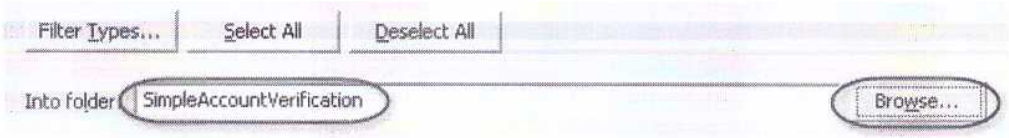
___h. In the **From directory** field, type **c:\lab**, then press the **Tab** key. Place a checkmark beside the **lab** folder in the list.



Hint: The business model defined in the previous part of this lab was exported to this directory. If this directory does not exist, or if the business model was not completed or exported successfully, then you can use the solution version of the business model at **c:\PoTFiles\lab** instead of **c:\lab**.



___i. Click on the **Browse** button to the right of the **Into folder** field. Select **SimpleAccountVerification** from the list, then click on **OK**.. It should then appear in the **Into folder** field.



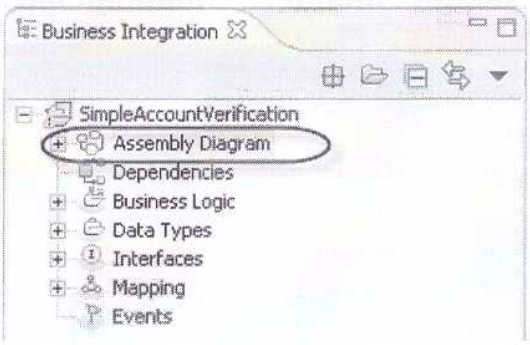
___j. Click on **Finish**. This will import the business model created earlier using the WebSphere Business Modeler.

___k. If a progress indicator appears, wait until it is complete. 

____ 2. Assemble the business process

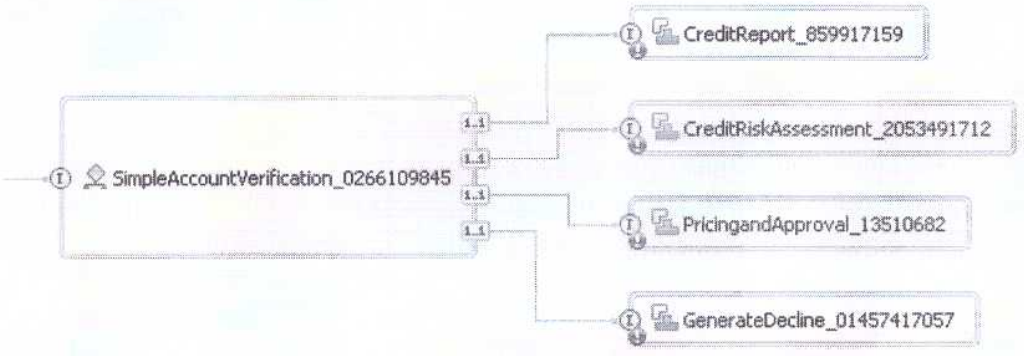
___a. By default, the **Assembly Diagram: SimpleAccountVerification** editor should already be open.

If not, then double-click on **SimpleAccountVerification -> Assembly Diagram** from the **Business Integration** view to open the Assembly Diagram editor.



The Assembly Diagram editor appears showing the different components of the business process. In IBM's SOA approach, the implementation of the tasks in the process is not tightly-coupled. For example, the implementation of the Credit Report task invoked by the SimpleAccountVerification process is not defined inside the process. It is kept separate and loosely-coupled, and is defined externally using the Assembly Diagram editor. This added level of abstraction provides the flexibility needed to quickly change implementations. This will be illustrated later in the lab.

The Assembly Diagram editor is where the building-block approach is applied. This editor allows you to integrate or assemble different components to compose a business process. Components can be web services, human tasks, BPEL processes, business rules, EJBs, Java programs, message-based (JMS) applications, or other legacy systems accessed using adapters or ESBs. These assemblies can integrate applications spanning different technologies, runtime platforms and programming languages. This is where we perform the "**assemble**" phase of the SOA cycle.



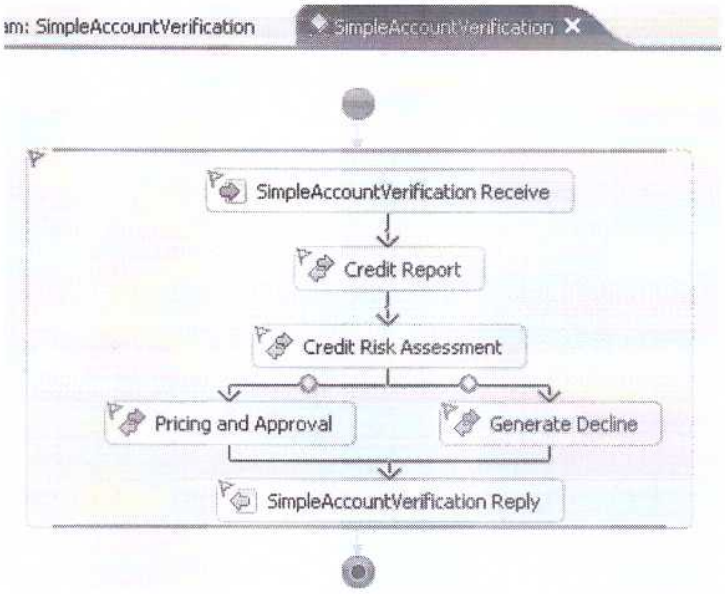
___b. Double-click on the **Assembly Diagram** editor tab to maximize the view.

Double-clicking on the tab again will restore the view back to its original size. For now, let's keep the view maximized so we can have more screen space to work with.

___c. From the **Assembly Diagram** editor, double-click on the **SimpleAccountVerification** process.

The SimpleAccountVerification process editor appears.

___d. Review the **SimpleAccountVerification** process.



This is the BPEL version of the business model defined using the WebSphere Business Modeler. BPEL will be explained later on. Because this was exported from the WebSphere Business Modeler, it is already complete and does not require further development.

___e. Close the **SimpleAccountVerification** process editor. 

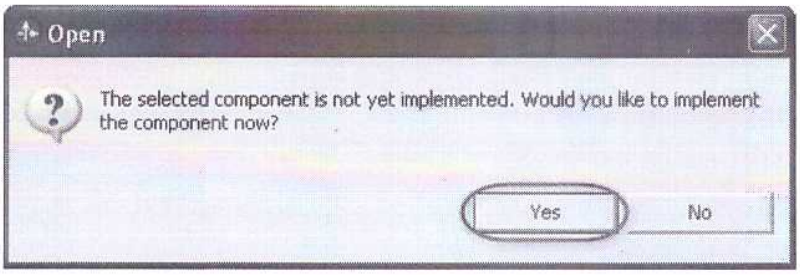
The **Assembly Diagram** editor will reappear. At this point, the tasks of the business process have been defined, as well as the flow through these tasks. However, the tasks have not yet been implemented. The next step is to implement the four tasks of our simple process.

These tasks will be initially implemented using Java code because the applications and back-end systems which need to be invoked by this process are still not yet accessible either through web services, messaging, adapters, or an enterprise service bus. That is still under development by other departments in your organization, or by business partners. For now, you will implement code stubs or placeholders for the actual applications and back-end systems until these become accessible.

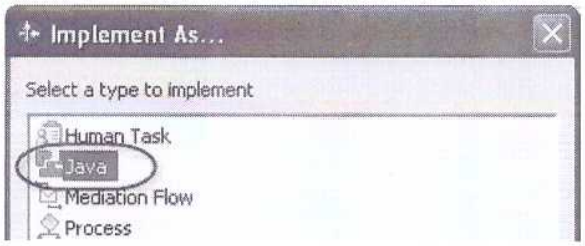
____ 3. Implement the tasks of the business process

___a. Double-click on the **Credit Report** component. 

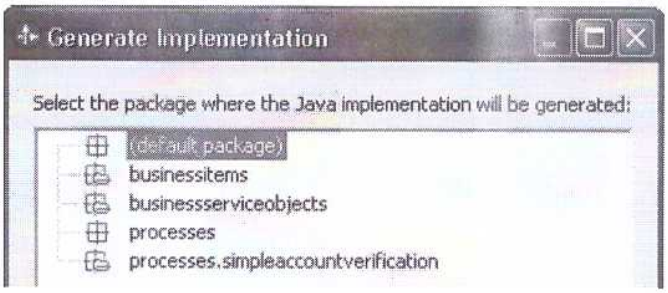
___b. From the **Open** confirmation window, click on **Yes**.



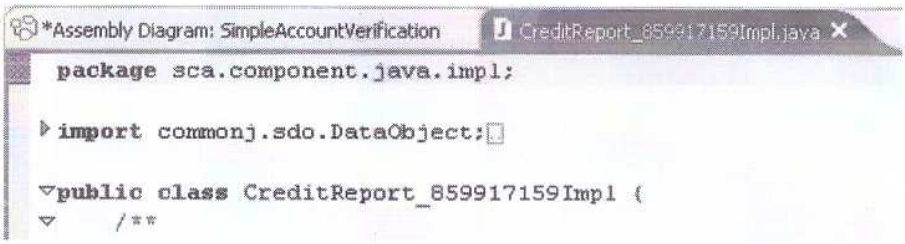
___c. From the **Implement As** window, select **Java** from the list, then click on **OK**.



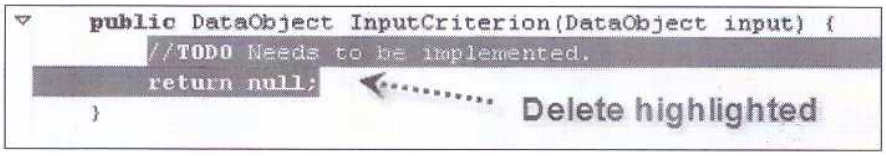
___d. From the **Generate Implementation** window, accept the default selection, then click on **OK**.



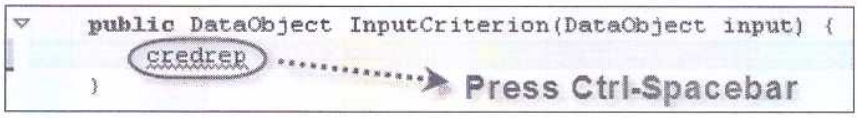
The Java editor appears for the implementation of the **CreditReport** component.



___e. Scroll down to the bottom of the Java code. Select the highlighted text below and **delete**.



___f. Type **credrep** in the line of code as shown below. Press **Ctrl-Spacebar**.



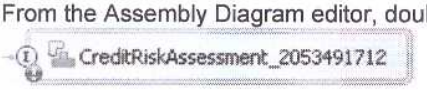
The **Ctrl-Spacebar** key combination activates a special feature called Code-Assist. This will add the required code snippet for you.

```
public DataObject InputCriterion(DataObject input) {
    /**
     *  CODE STUB
     *  Placeholder for production code later
     *  Code only for initial development and testing
     */
    int creditScore = 501;
    DataObject customer = input.getDataObject("customer");
    if (customer.getString("customerID").equalsIgnoreCase("123"))
        creditScore = 449;
    System.out.println(">>>>>>>>>>>>>>>>>>>>>>");
    System.out.println(">>>>> Credit Score = " + creditScore);
    input.setInt("creditScore", creditScore);
    return input;
}
```

This code logic will return a credit score of 449 if the Customer ID submitted in the request is "123", otherwise a credit score of 501 will be returned. A credit score below 500 will be considered high-risk, and a score above 500 will be considered low-risk. In other words, a request with a Customer ID of "123" will be rejected, and any other Customer ID will be approved.

___g. Press **Ctrl-s** to save the Java code, then close the Java editor.

___h. From the Assembly Diagram editor, double-click on the **Credit Risk Assessment** component.

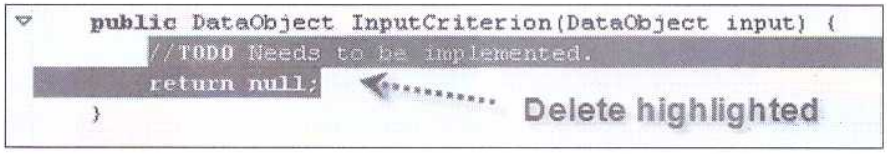


___i. From the **Open** confirmation window, click on **Yes**.

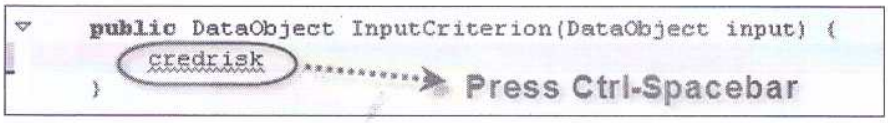
___j. From the **Implement As** window, select **Java** from the list, then click on **OK**.

___k. From the **Generate Implementation** window, accept the default selection, then click on **OK**.

___l. In the Java editor, scroll down and perform the following code modifications:



Type **credrisk**.

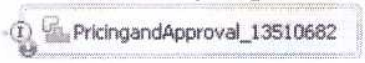


```
public DataObject InputCriterion(DataObject input) {
    /**
     *  CODE STUB
     *  Placeholder for production code later
     *  Code only for initial development and testing
     */
    String riskAssessment = "LOW";
    if (input.getInt("creditScore") < 500)
        riskAssessment = "HIGH";
    System.out.println(">>>>>>>>>>>>>>>>>>>>>>");
    System.out.println(">>>>> Credit risk = " + riskAssessment);
    input.setString("creditRiskAssessment", riskAssessment);
    return input;
}
```

This code will assign a risk assessment of "LOW" if the credit score is above 500. Conversely, a credit score below 500 will be assigned a risk assessment of "HIGH".

___m. Press **Ctrl-s** to save the Java code, then close the Java editor.

___n. From the Assembly Diagram editor, double-click on the **Pricing and Approval** component.

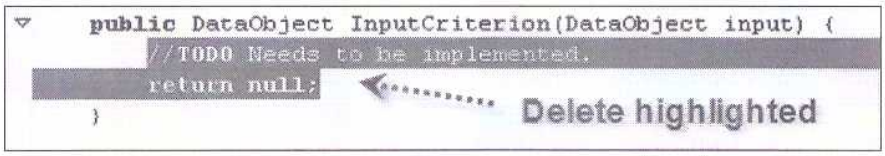


___o. From the **Open** confirmation window, click on **Yes**.

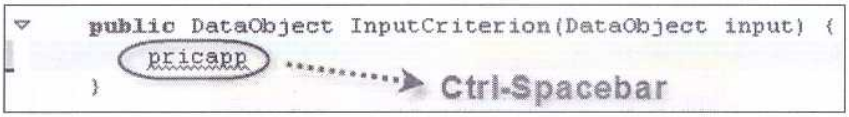
___p. From the **Implement As** window, select **Java** from the list, then click on **OK**.

___q. From the **Generate Implementation** window, accept the default selection, then click on **OK**.

___r. In the Java editor, scroll down and perform the following code modifications:



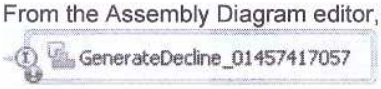
Type **pricapp**.



```
public DataObject InputCriterion(DataObject input) {
    /**
     *  CODE STUB
     *  Placeholder for production code later
     *  Code only for initial development and testing
     */
    System.out.println(">>>>> Request is approved!.");
    return input;
}
```

___s. Press **Ctrl-s** to save the Java code, then close the Java editor.

___t. From the Assembly Diagram editor, double-click on the **Generate Decline** component.

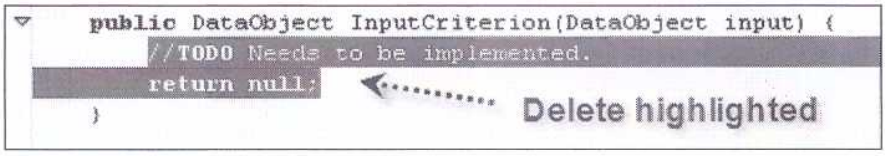


___u. From the **Open** confirmation window, click on **Yes**.

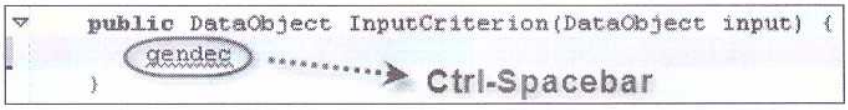
___v. From the **Implement As** window, select **Java** from the list, then click on **OK**.

___w. From the **Generate Implementation** window, accept the default selection, then click on **OK**.

___x. In the Java editor, scroll down and perform the following code modifications:



Type **gendec**.




```
public DataObject InputCriterion(DataObject input) {
    /**
     *  CODE STUB
     *  Placeholder for production code later
     *  Code only for initial development and testing
     */
    System.out.println(">>>>> Request is denied!.");
    return input;
}
```

___y. Press **Ctrl-s** to save the Java code, then close the Java editor.

___z. From the **Assembly Diagram** editor, press **Ctrl-s** to save your work. Do not close the **Assembly Diagram** editor because this will be needed for later steps.

The **"Assemble"** phase of our SOA approach to implementing this simple business process is now complete.

Proceed to the next part.

Part 4: "Deploy" the Business Process

____ 4. Deploy and test the business process

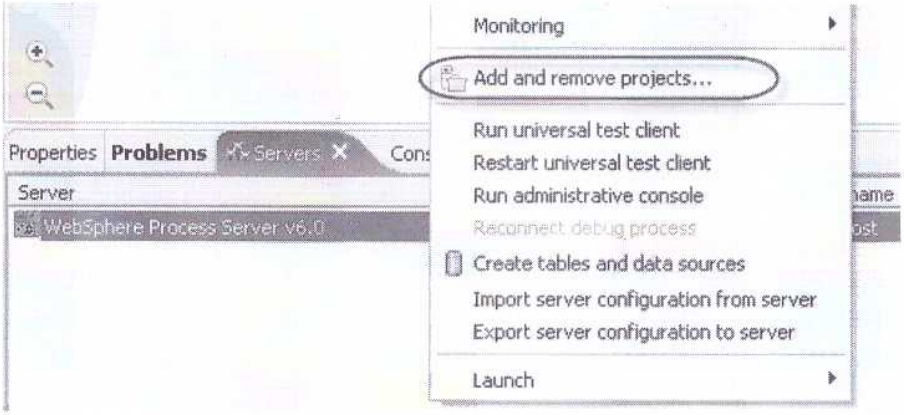
___a. Double-click on the **Assembly Diagram** editor tab to restore the view to its original size.

___b. In the bottom section, click on the **Problems** tab. Ensure that there are no errors listed.

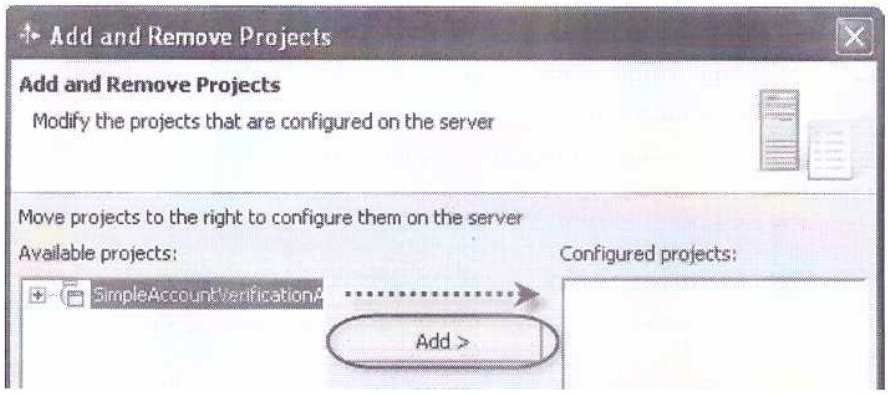
___c. Click on the **Servers** tab. Verify that the **WebSphere Process Server v6.0** test server is already started.

This test server built into the WebSphere Integration Developer is a full instance of the **WebSphere Process Server**.

___d. Right-click on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Add and remove projects**.

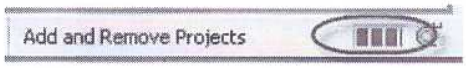


___e. From the **Add and Remove Projects** window, click on **Add**.

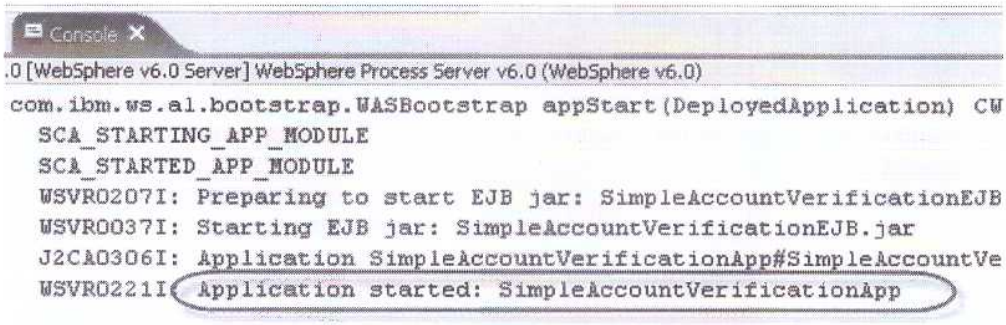


___f. Click on **Finish**. This will add the business process to the test server configuration.

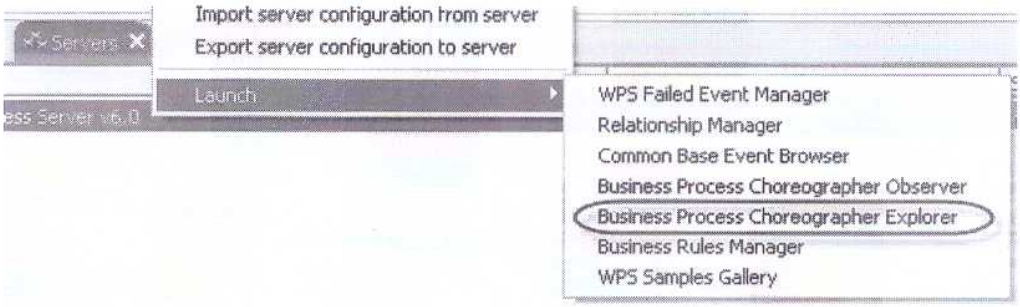
___g. A progress indicator will appear in the lower-right corner of the WebSphere Integration Developer window. Wait for this indicator to disappear before proceeding to the next step.



___h. Focus will automatically switch to the **Console** view. Verify that a message appears in the **Console** view indicating that the **SimpleAccountVerificationApp** process has started. Scroll right as needed.

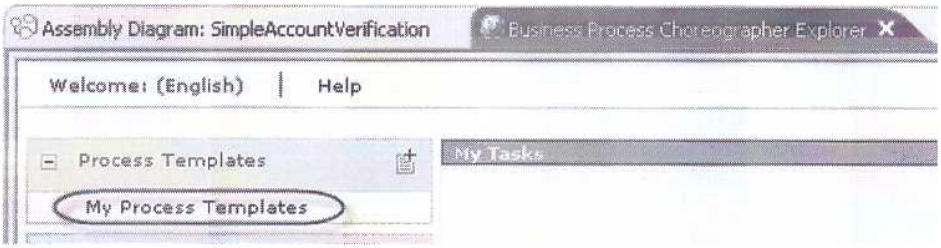


___i. Switch to the **Servers** view and right-click on **WebSphere Process Server v6.0**. From the popup menu, select **Launch -> Business Process Choreographer Explorer**.

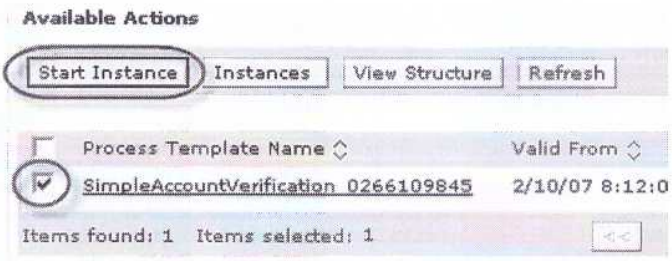


The Business Process Choreographer Explorer appears. This is a built-in web application that provides a way to manage business processes, as well as work items associated with human tasks in the process.

___j. Click on the **My Process Templates** link to display a list of processes which can be started.

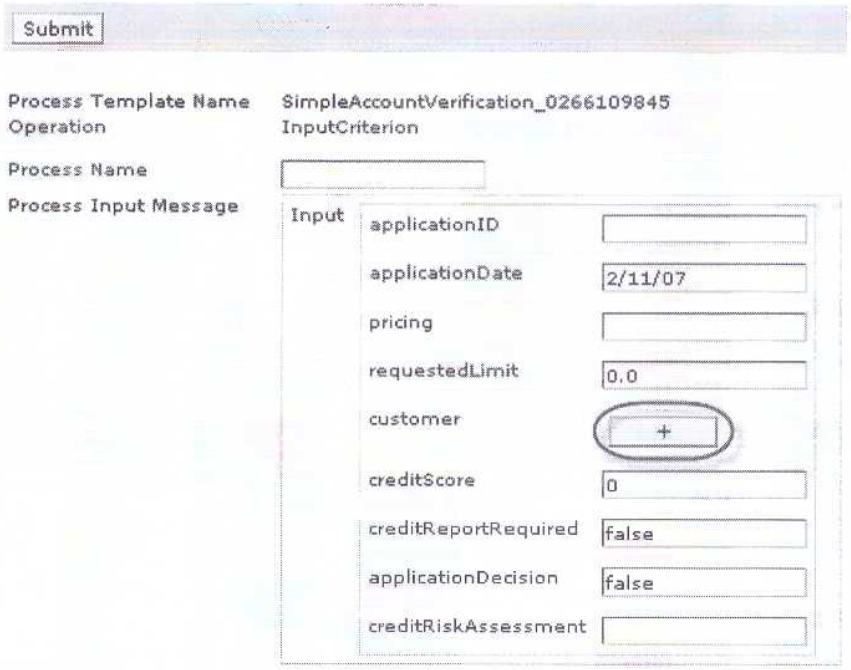


___k. Place a checkmark beside the **SimpleAccountVerification** process template, then click on **Start Instance**.

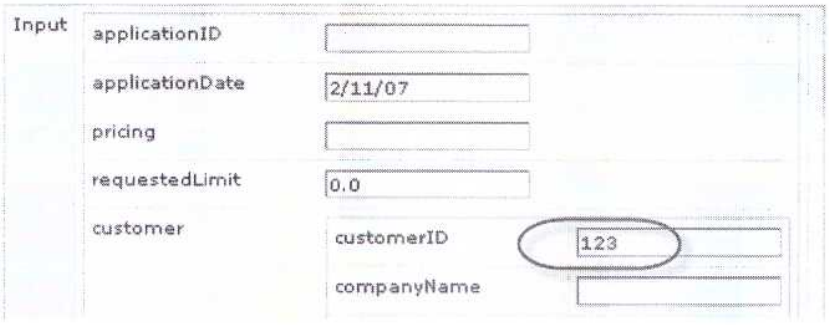


This will start the SimpleAccountVerification business process, and display the default input page.

___l. From the default input page, click on the Expand button.

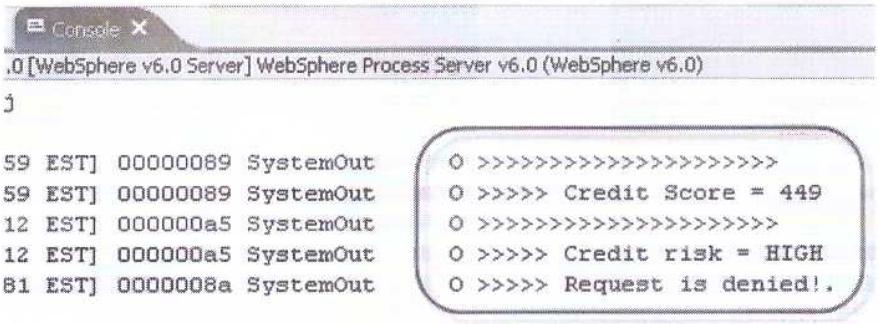


___m. For the **customerID** field, type **123**, accept the defaults for the other fields, then click on **Submit**.

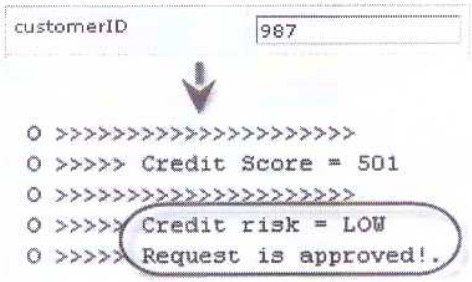


A Customer ID of "**123**" will result in a low credit score and a denied request. Any other Customer ID value will result in an approved request.

___n. After a few seconds, the SimpleAccountVerification process will generate messages to the **Console** view indicating that the Request is denied.



This indicates that the business process is functioning properly. You can also try other input values for the customerID to see the request approved.



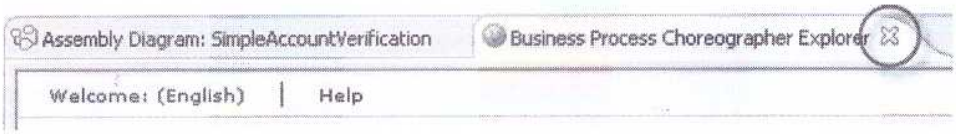
___o. When you're done testing the business process, switch to the **Servers** view.

___p. Right-click on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Add and remove projects**.

___q. From the **Add and Remove Projects** window, click on **Remove All**.

___r. Click on **Finish**.

___s. Close the **Business Process Choreographer Explorer**.



The **"Deploy"** phase of our SOA approach to implementing this simple business process is now complete.

Please wait for the next lecture before proceeding with the lab.

Part 5: Handling Change with IT Flexibility

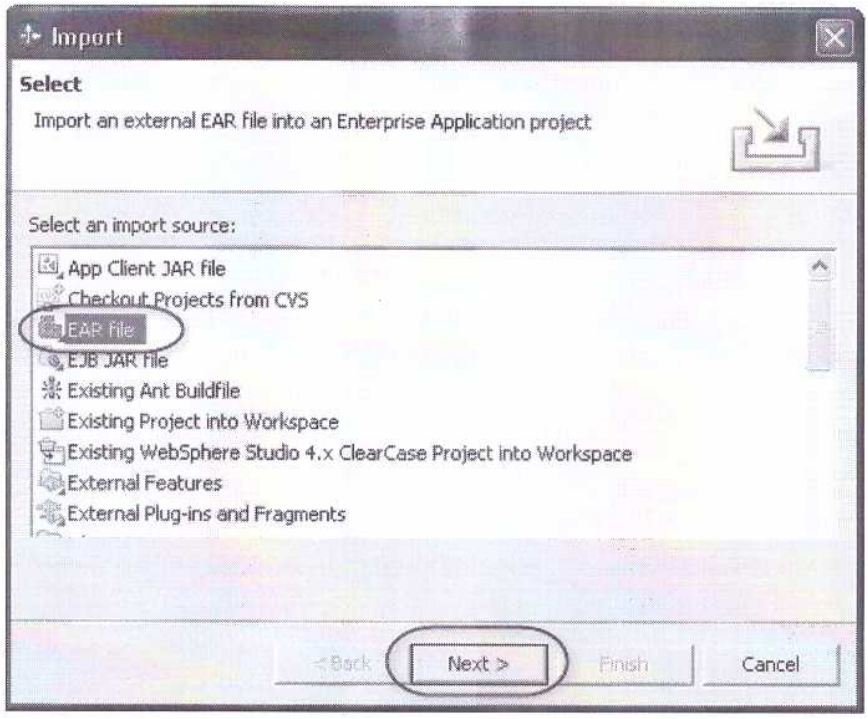
To illustrate the IT flexibility needed for business flexibility, you will change the implementation of one of the tasks without major effort. By employing the building block approach and loose-coupling, switching to a different application or system becomes easier and quicker, even though the new system may involve a different technology, programming language, communication mechanism, and runtime environment. This will allow business processes to be more responsive to changes in their business environment. These changes might involve having to quickly adapt to customer demands, exploit new market opportunities, or react to competitive threats. SOA provides this type of business agility.

In our simple scenario, let's assume that the Credit Report task now needs to invoke an existing system exposed as a web service. You will now change the implementation type of this task from a Java program to a web service invocation. The web service is really just an example in this exercise. Instead of a web service, you can just as easily switch the Credit Report task to a human task, business rule, business process, etc.

____ 1. Import the Credit Report Web Service

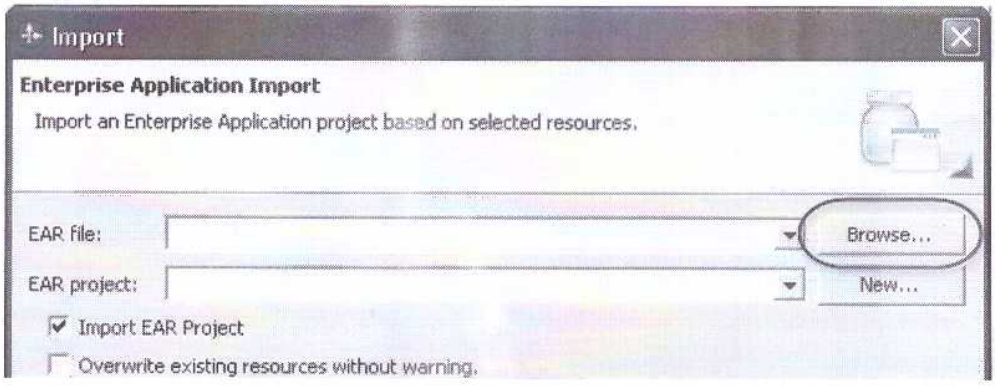
___a. From the main menu, select **File -> Import**.

___b. From the **Import** window, select **EAR file**, then click on **Next**.

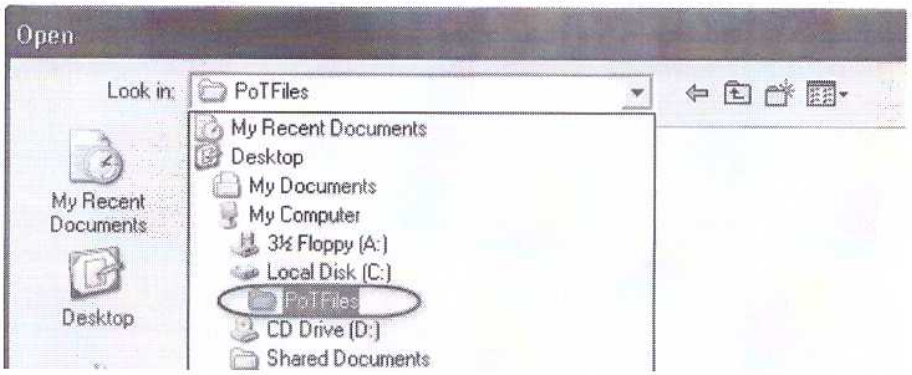


The Credit Report web service will need to be imported as an Enterprise Archive file (EAR). The EAR file contains all the code and artifacts needed to run the web service. In a production environment, you typically only need the web service description file, or the WSDL file, to invoke the web service because the web service code should be running on a separate remote server. However, for this lab, we will deploy and run the web service in the test server of this development environment.

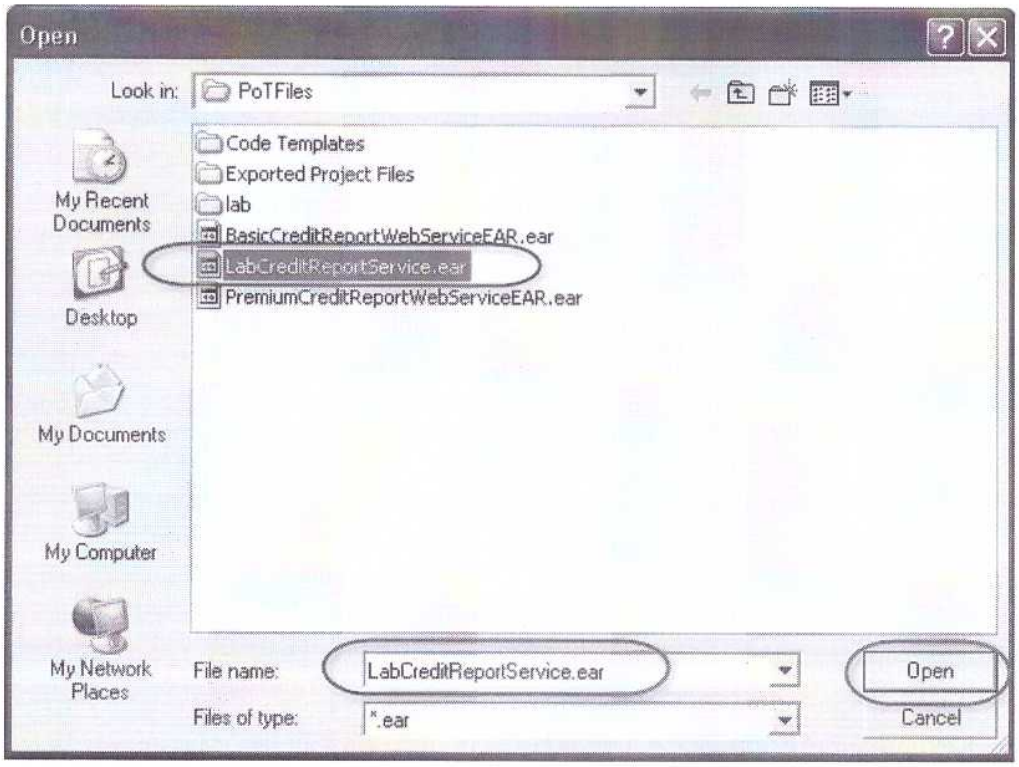
___c. From the **Import** window, click on the **Browse**.



___d. From the **Open** window, using the drop-down list, navigate to the directory **C:\PoTFiles**.

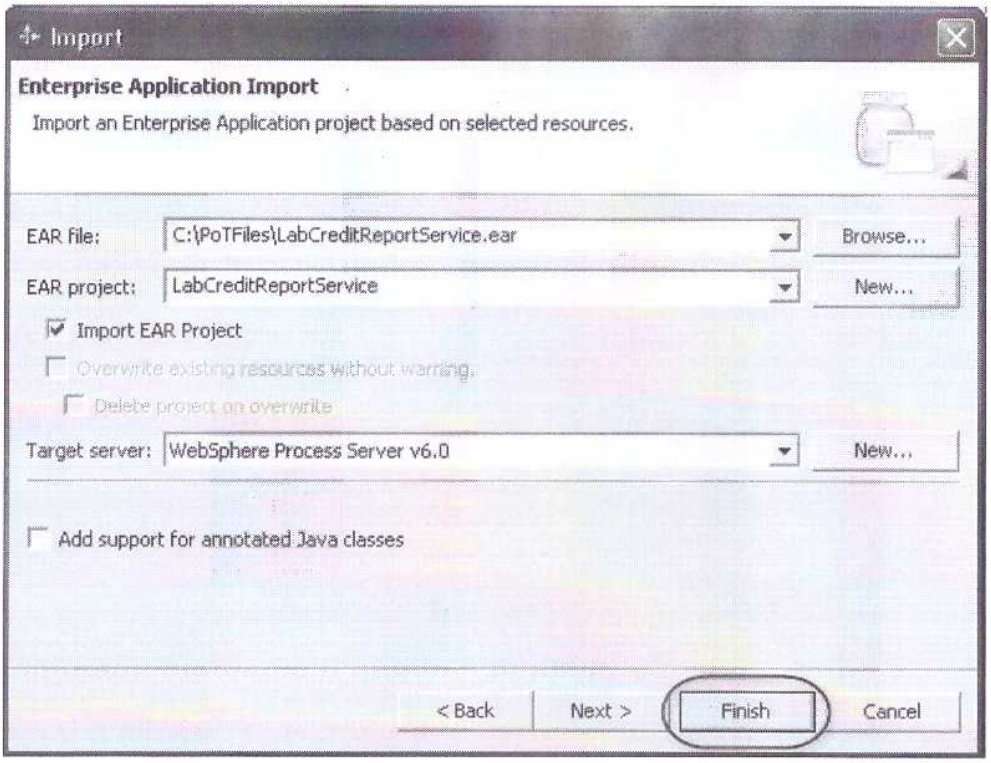


___e. From the **Open** window, select the **LabCreditReportService.ear** file, then click on **Open**.



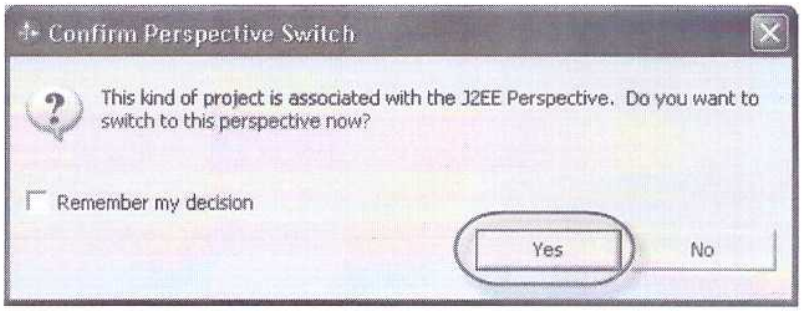
This is the EAR file containing the Credit Report web service.

___f. Click on **Finish**.



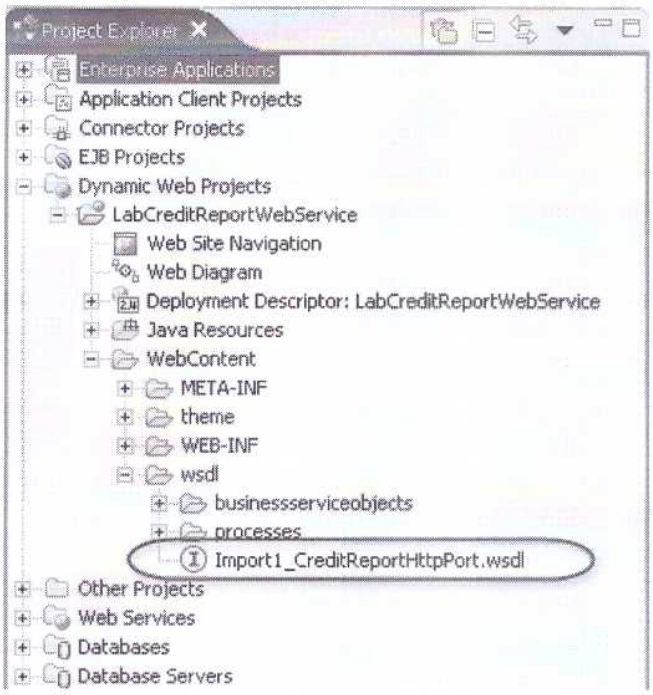
This will load all the code and artifacts of the CreditReport web service into the current workspace of the WebSphere Integration Developer.

___g. From the **Confirm Perspective Switch** window, click on **Yes**.



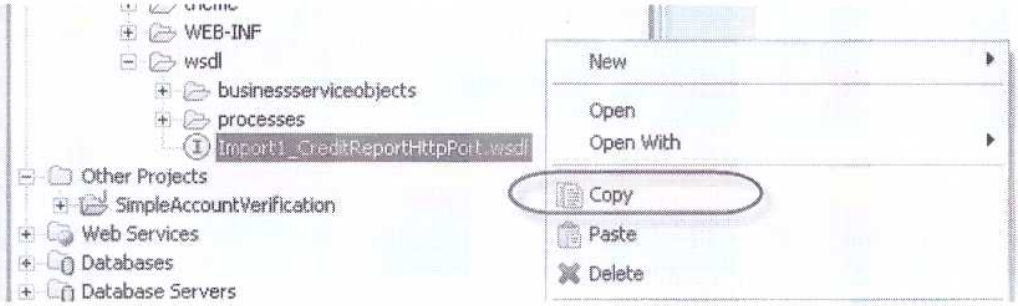
This will switch to the **J2EE** perspective to allow us to view the Credit Report web service. You were previously in the Business Integration perspective, which is better-suited for your role as an Integration Developer. We will return to the Business Integration perspective shortly.

___h. Expand the tree view to select **Dynamic Web Projects -> LabCreditReportWebService -> WebContent -> wsdl -> Import1_CreditReportHttpPort.wsdl**.



This is the description file for the **Credit Report** web service which will be invoked by the **SimpleAccountVerification** process.

___i. Right-click on the **Import1_CreditReportHttpPort.wsdl** file. From the popup menu, select **Copy**.

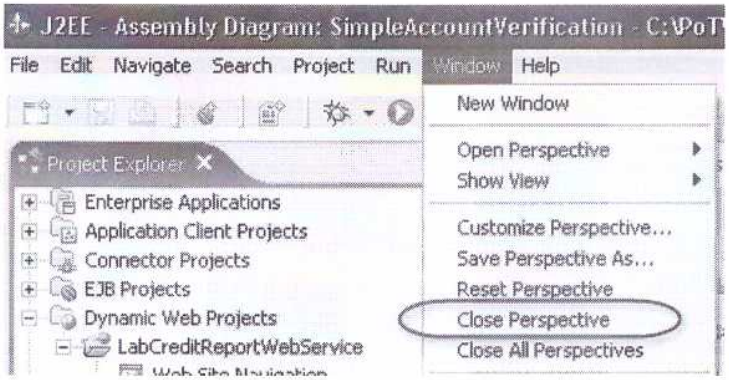


___j. From the tree view, expand the **Other Projects** folder, and right-click on the **SimpleAccountVerification** folder.

This will copy the WSDL file for the **CreditReport** web service into the

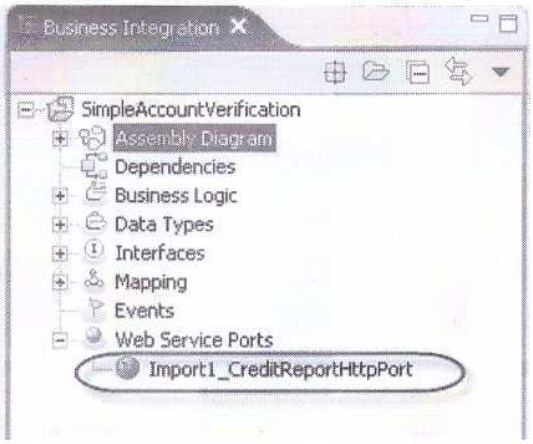
SimpleAccountVerification project. This will allow us to visually connect this web service to the SimpleAccountVerification process in the Assembly Diagram editor. In a realistic development project, you probably won't need to copy the WSDL file because this will typically be loaded from an external registry, such as the WebSphere Service Registry and Repository or a UDDI registry.

___k. From the main menu, select **Window -> Close Perspective**.



This will close the **J2EE** perspective and return focus to the **Business Integration** perspective. The top right corner of the main window displays the current perspective.

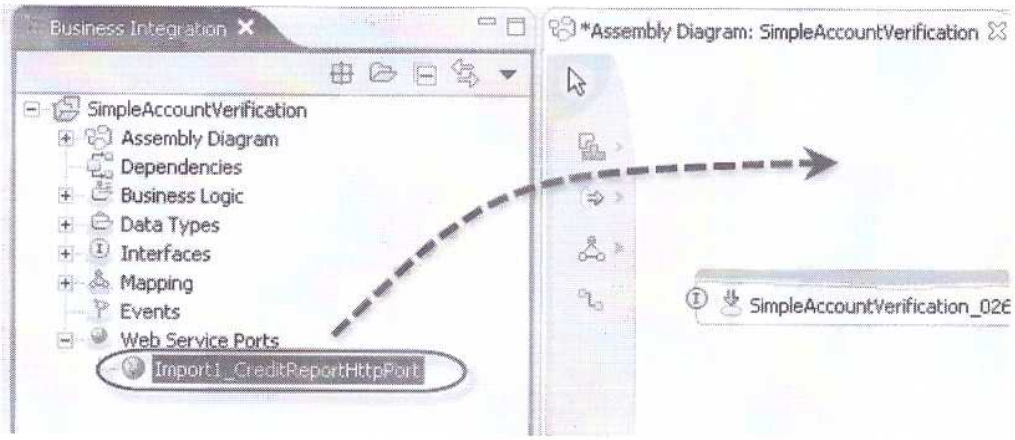
___l. In the Business Integration view, verify that the Credit Report web service appears under Web Service Ports.



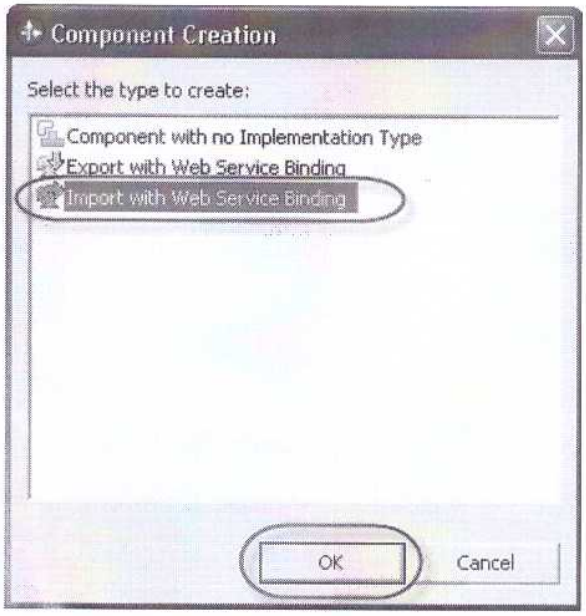
____ 2. Change the Credit Report task to a Web Service Implementation

___a. From the **Assembly Diagram** editor, select the **CreditReport** component, then press the **delete** key.

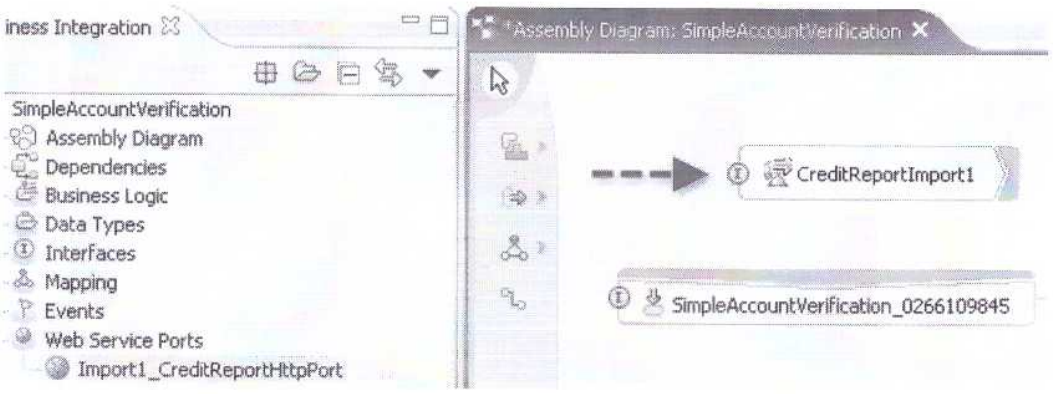
___b. From the tree view, select **Import1_CreditReportHttpPort**. Using the left mouse button, drag and drop **Import1_CreditReportHttpPort** into an empty space in the **Assembly Diagram** editor.



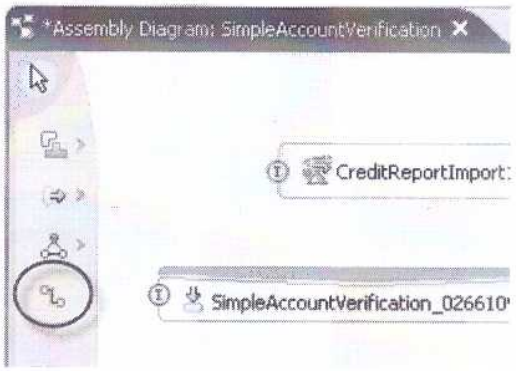
___c. Select **Import with Web Service Binding**, then click on **OK**.



A **CreditReport** Web Service component will appear in the **Assembly Diagram** editor.



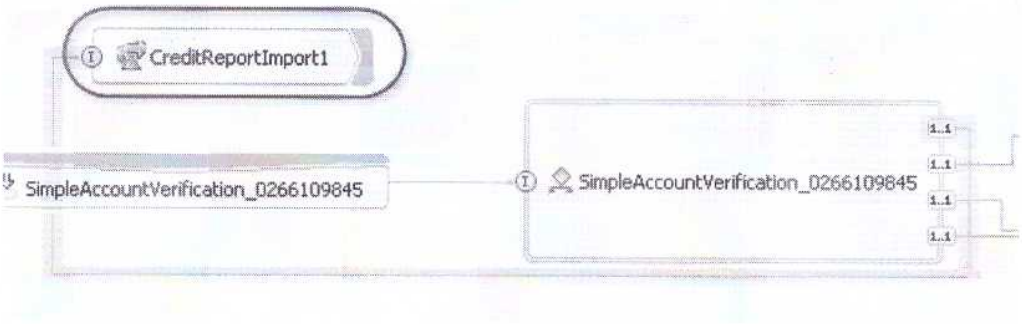
___d. From the palette on the left, select the **Wire** icon.



This will switch the editor to 'wire' mode, which will make it easier to connect components in the Assembly Diagram editor.

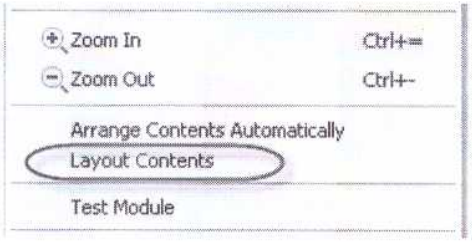
___e. Left-click on the CreditReport reference point on the SimpleAccountVerification process component. This will start a connection (blue line).

___f. Click on the **CreditReportImport1** component. This will complete the new connection highlighted in yellow below.



___g. From the palette, click on the **Selection Tool** icon to switch back to selection mode.

___h. Right-click on an empty space in the **Assembly Diagram** editor. From the popup menu, select **Layout Contents**.



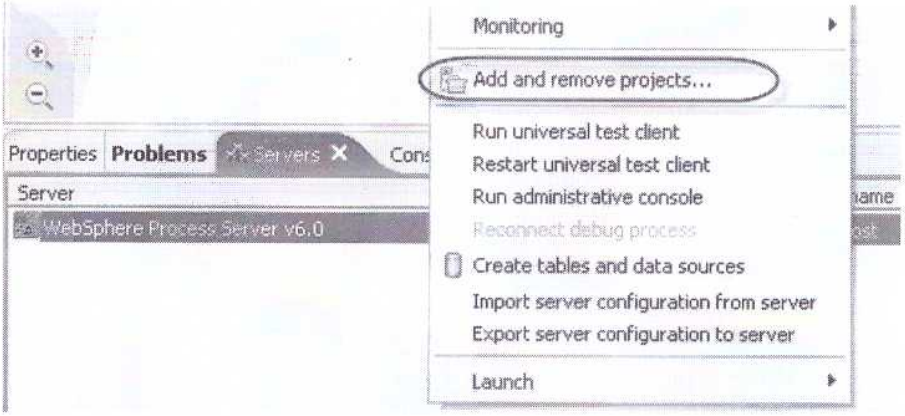
This will rearrange the **CreditReportImport1** component for a cleaner layout.

___i. Press **Ctrl-s** to save the modified assembly.

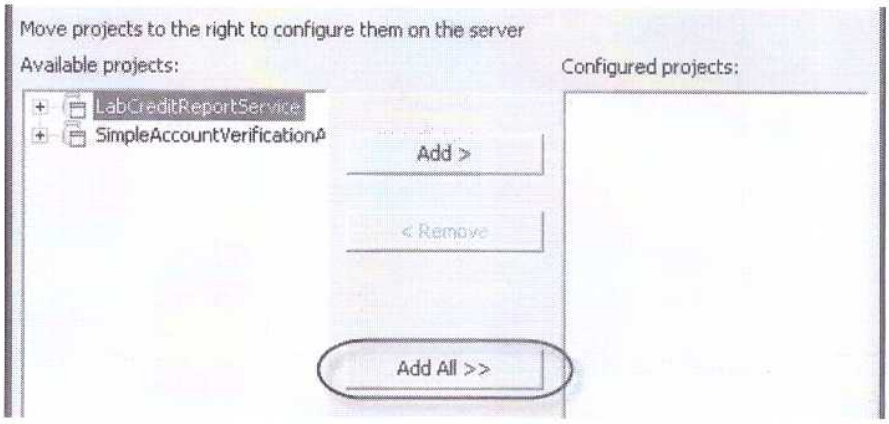
____ 3. Retest the SimpleAccountVerification Process

___a. Switch to the **Servers** view.

___b. Right-click on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Add and remove projects**.



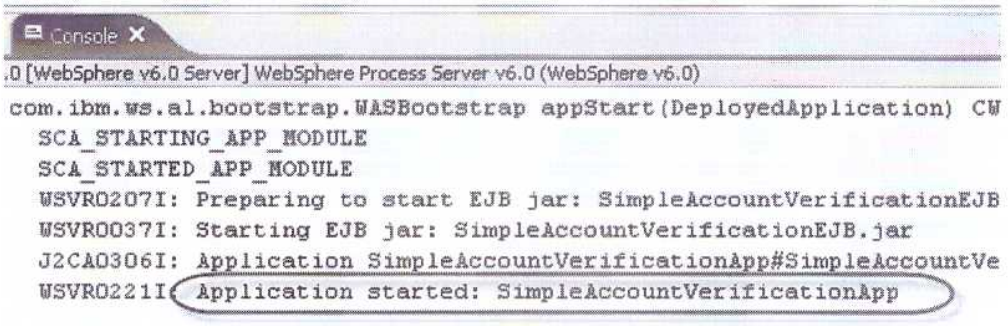
___c. From the **Add and Remove Projects** window, click on **Add All**.



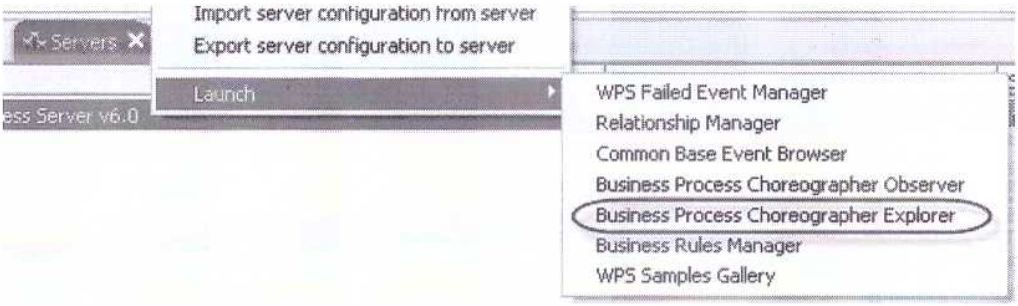
___d. Click on **Finish**. This will add the business process to the test server configuration.

___e. A progress indicator will appear in the lower-right corner of the WebSphere Integration Developer window. Wait for this indicator to disappear before proceeding to the next step.

___f. Focus will automatically switch to the **Console** view. Verify that a message appears in the **Console** view indicating that the **SimpleAccountVerificationApp** process has started. Scroll right as needed.



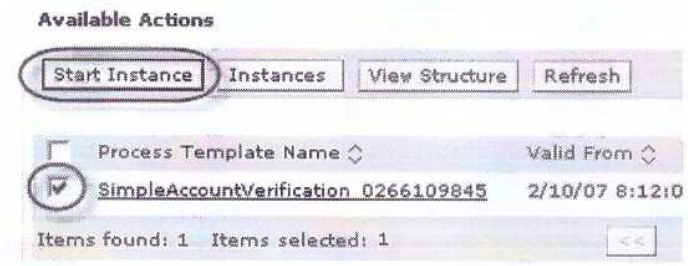
___g. Switch to the **Servers** view and right-click on **WebSphere Process Server v6.0**. From the popup menu, select **Launch->Business Process Choreographer Explorer**.



The Business Process Choreographer Explorer appears. This is a built-in web application that provides a way to manage business processes, as well as work items associated with human tasks in the process.

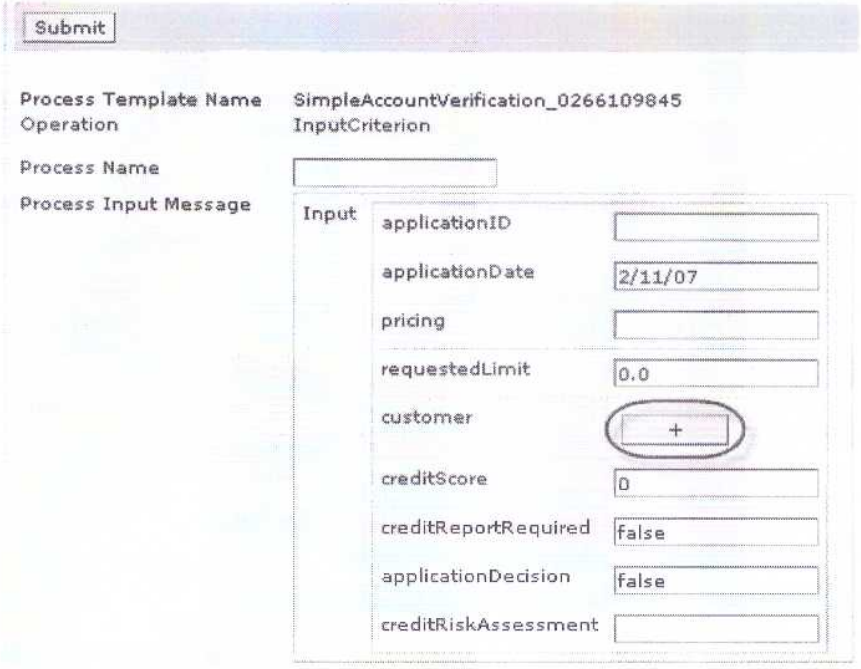
___h. Click on the **My Process Templates** link to display a list of processes which can be started.

___i. Place a checkmark beside the **SimpleAccountVerification** process template, then click on **Start Instance**.

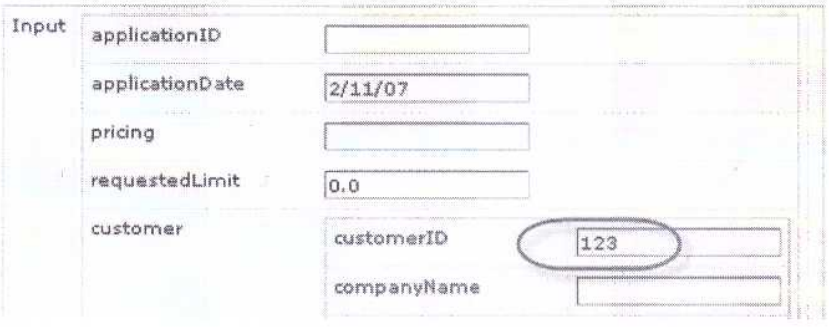


This will start the SimpleAccountVerification business process, and display the default input page.

___j. From the default input page, click on the Expand button.



___k. For the **customerID** field, type **123**, accept the defaults for the other fields, then click on **Submit**.

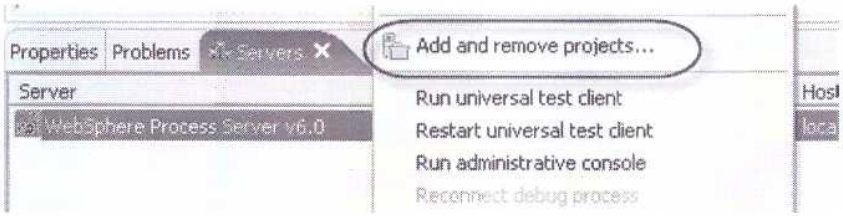


A Customer ID of "**123**" will again result in a low credit score and a denied request. Any other Customer ID value will result in an approved request. However, this time the messages will indicate that a web service is invoked instead of the previous Java code.

```
Fram A   CHFW0019I: The Transport Channel
appe A   SRVE0242I: [LabCreditReportServic
   O **********************
   O * Call Web Service   *
   O **********************
   O >>>>>>>>>>>>>>>>>>>>>
   O >>>>> Credit Score = 449
   O >>>>>>>>>>>>>>>>>>>>>
   O >>>>> Credit risk = HIGH
   O >>>>> Request is denied!.
```

___l. When you're done testing the business process, switch to the **Servers** view.

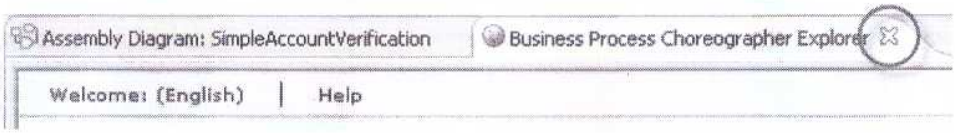
___m. Right-click on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Add and remove projects**.



___n. From the **Add and Remove Projects** window, click on **Remove All**.

___o. Click on **Finish**.

___p. Close the **Business Process Choreographer Explorer**.



Please wait for the next lecture before proceeding with the lab.

Part 6: Business Rules

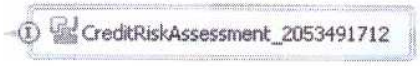
Changes in your business environment often involve the need to adjust business variables such as discount rates, fees, or product pricing based on promotions, time of year, etc. It might also involve modifications to business algorithms or decision tables. Business processes need to be able to quickly adapt to these changes, without time, cost, and resource-intensive development and maintenance work.

The Business Rules capability of the WebSphere Process Server can provide that type of business flexibility, where business variables and formulas can be loosely-coupled or externalized from the business processes. This feature allows for quick and easy modifications to these business rules without recoding, redeployment, or retesting.

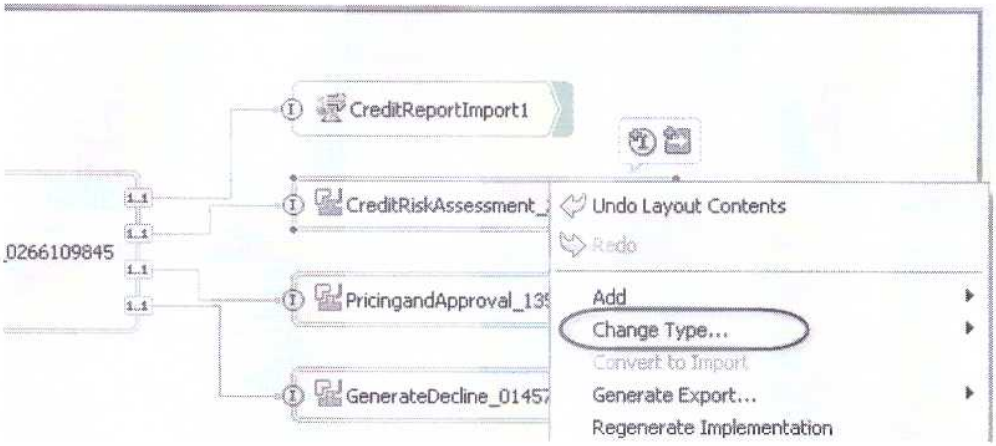
The Business Rules feature of the WebSphere Process Server also adheres to the building-block approach, where these business rules are modularized and treated just like any other service component. This will be illustrated in this part of the lab where you will quickly change the Credit Risk Assessment task from a Java implementation to Business Rules.

____ 1. Change the Credit Risk Assessment Task to a Business Rule

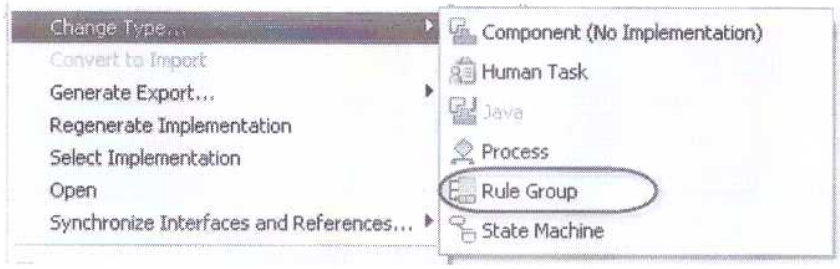
___a. From the **Assembly Diagram** editor, right-click on the **CreditRiskAssessment_2053491712** component.



___b. From the popup menu, select **Change Type**.



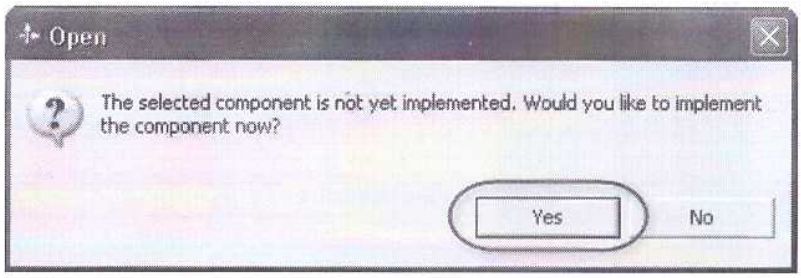
___c. From the secondary popup menu, select **Rule Group**.



This will change the implementation type from Java to a Rule Group. The logic to determine whether the risk assessment rating is HIGH or LOW, used to be in Java code. It will now be replaced by a Business Rule for better flexibility.

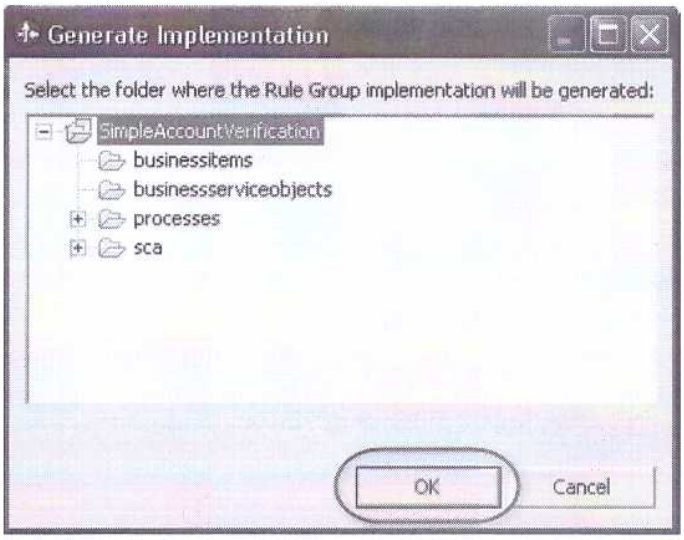
___d. Double-click on the **CreditRiskAssessment_2053491712** component.

___e. From the **Open** window, click on **Yes**.



An empty Business Rule Group will be created. The next step is to define the specific rules to determine if the credit risk assessment rating should either be HIGH or LOW based on the credit score.

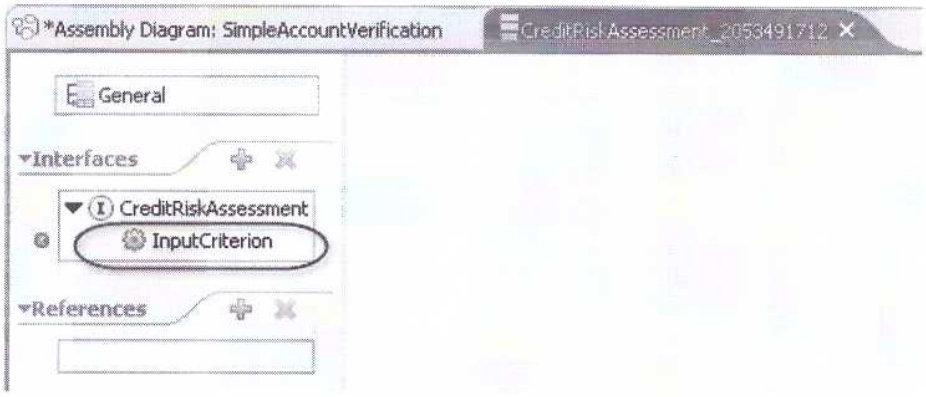
___f. From the **Generate Implementation** window, click on **OK**.



The Business Rule Group editor appears.

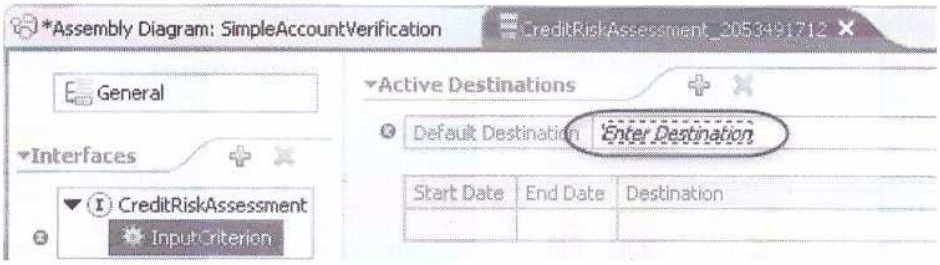
____ 2. Define the Business Rule Set

___a. From the **Business Rule Group** editor, click on the **InputCriterion** Interface.

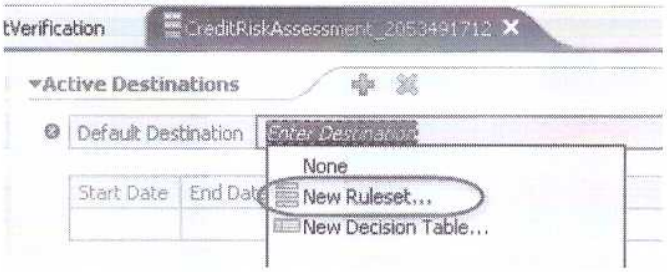


A Business Rule Group contains related Business Rules. In other words, A Business Rule Group is a logical grouping of related Rules.

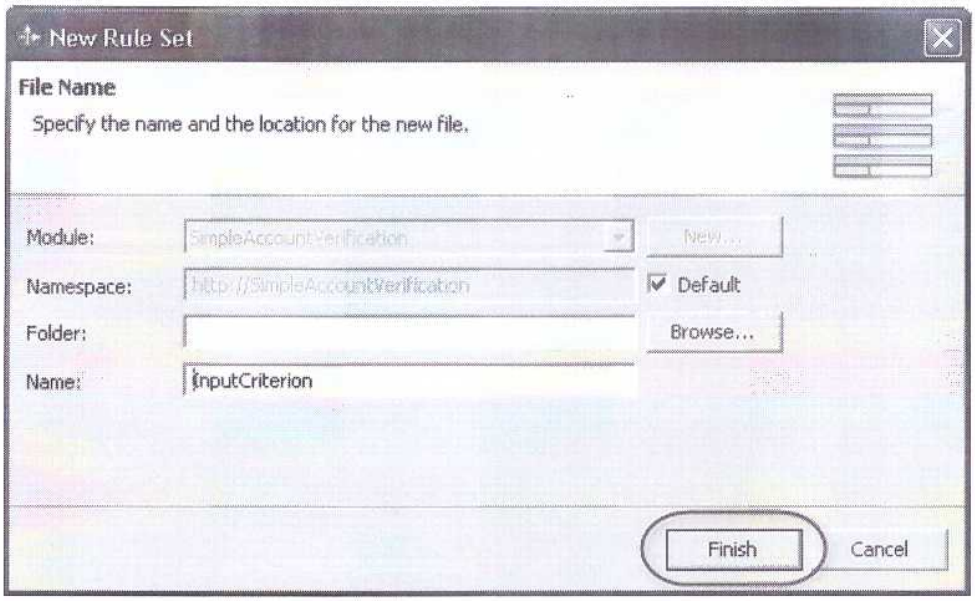
___b. Click on the **Enter Destination** link.



___c. Select **New Ruleset** from the popup list.

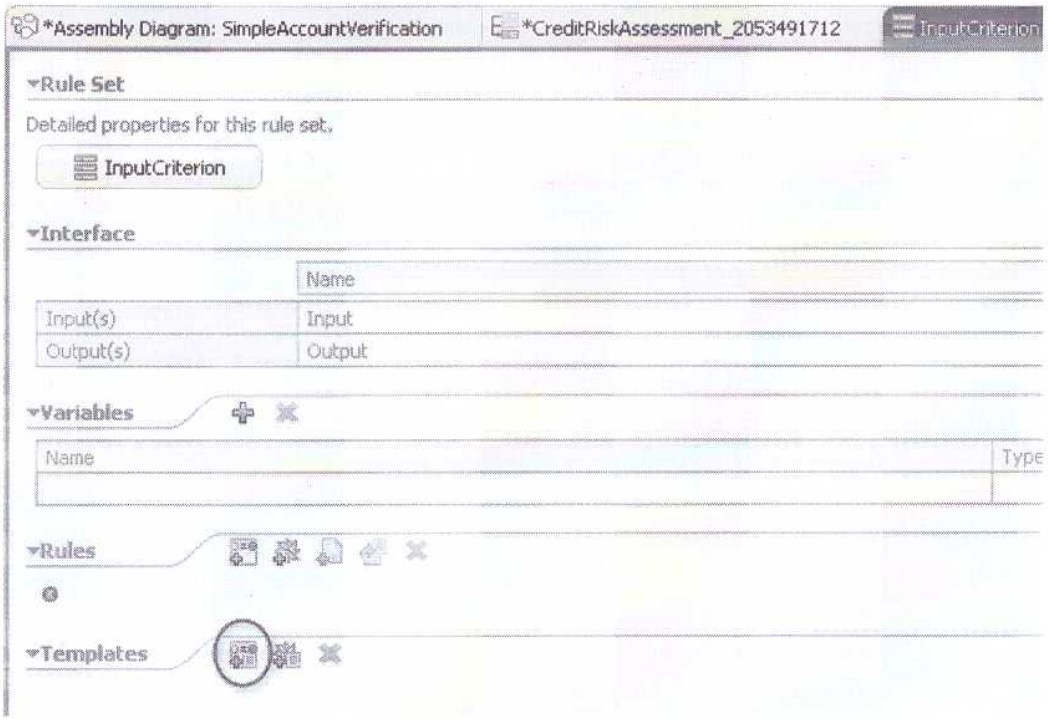


___d. From the **New Rule Set** window, accept the defaults and click on **Finish**.



The Rule Set editor appears.

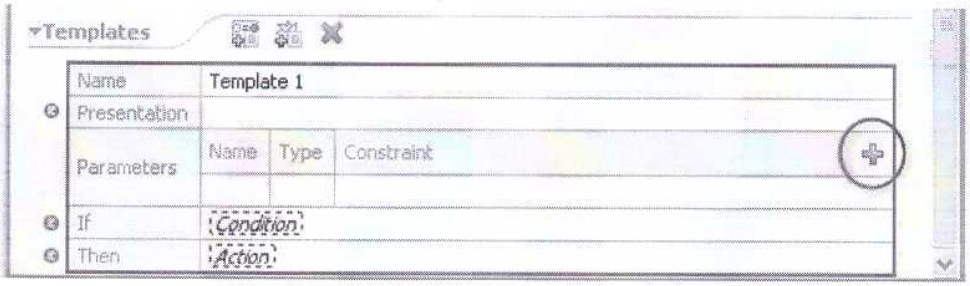
___e. From the InputCriterion Rule Set editor, click on the **Add If Then Template** icon.



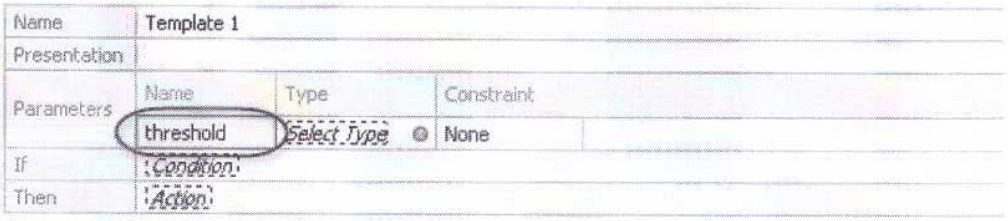
The next step is to define a business rule where the risk assessment will be rated "HIGH" if the credit score from the CreditReport web service is less than a certain threshold. In the previous

Java implementation, the threshold was fixed at 500. In this business rule, it will be a variable that can be changed at runtime.

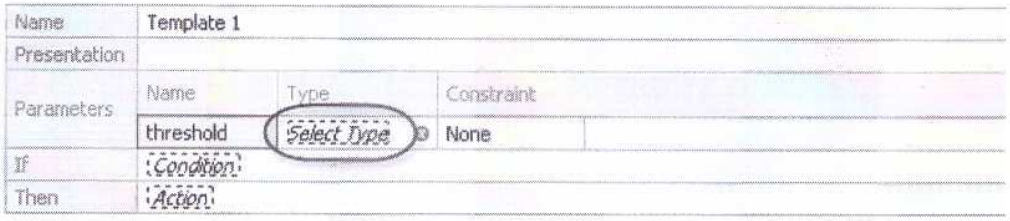
___f. In the **Templates** section of the Rule Set editor, click on the **Add Template Parameter** icon.



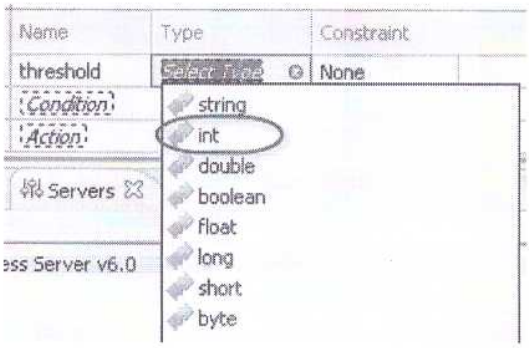
___g. Change the new Parameter Name from **param1** to ***threshold***.



___h. Click on the **Select Type** link.

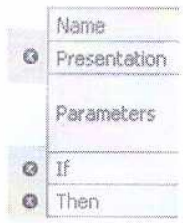


___i. From the popup list, select **int**.

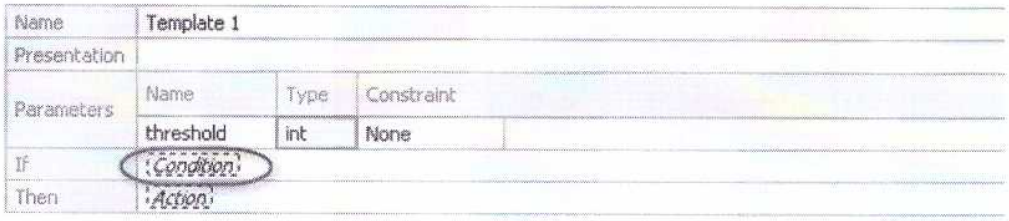


Take note of the red 'x' marks (⊗) highlighted below. This indicates that an error exists in that specific field. That is to be expected at this point because the proper values still have to be

specified.

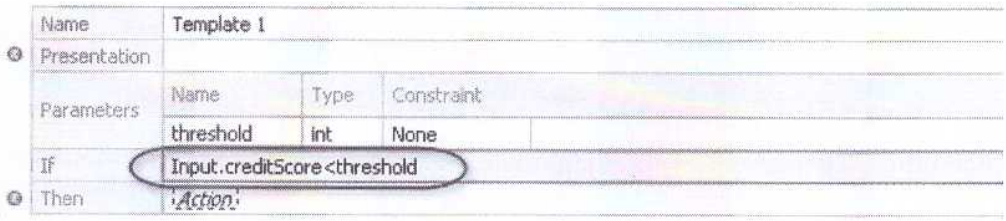


___j. Click on the **Condition** link in the **If** field.



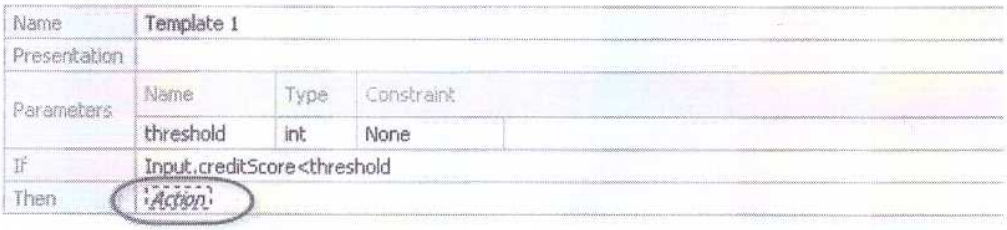
___k. Ignore the popup list and just type the following into the **If** field:

Input.creditScore<threshold



The red 'x' mark (⊗) beside the **If** field should disappear if the Condition value was specified correctly.

___l. Click on the **Action** link in the **Then** field.



___m. Ignore the popup list and type the following into the **Then** field:

Output.creditRiskAssessment="HIGH"

Name	Template 1		
Presentation			
Parameters	Name	Type	Constraint
	threshold	int	None
If	Input.creditScore<threshold		
Then	Output.creditRiskAssessment="HIGH"		

The red 'x' mark ( ) beside the **Then** field should disappear if the Condition value was specified correctly.

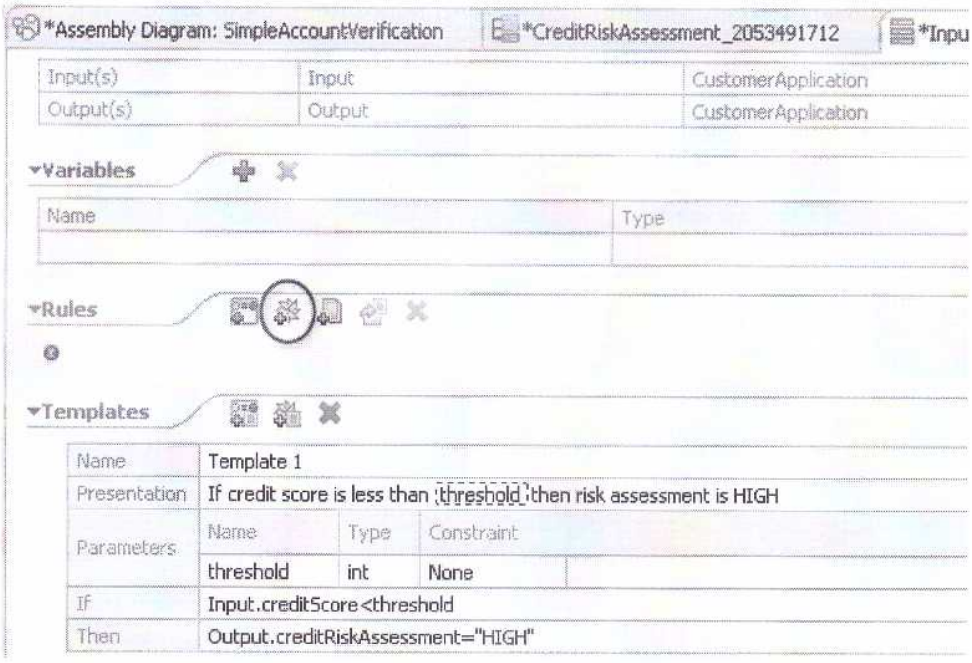
___n. Type the following into the **Presentation** field:

If credit score is less than {threshold} then risk assessment is HIGH

Name	Template 1		
Presentation	If credit score is less than {threshold} then risk assessment is HIGH		
Parameters	Name	Type	Constraint
	threshold	int	None
If	Input.creditScore<threshold		
Then	Output.creditRiskAssessment="HIGH"		

The text specified in the Presentation field will be displayed in the Business Rules Manager user interface. The Business Rules Manager will allow you to dynamically change the parameter values in the template rule while the process is running. In this case, the ***threshold*** parameter can be changed at runtime to affect how the risk assessment rating is determined, without having to modify, retest, and redeploy the process. This will be illustrated later in the lab.

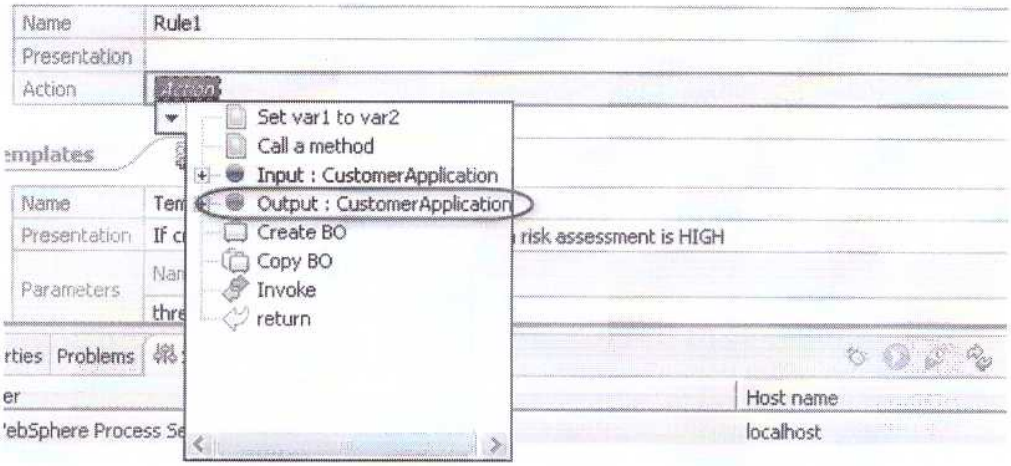
___o. In the **Rules** section, click on the **Add Action Rule** icon.



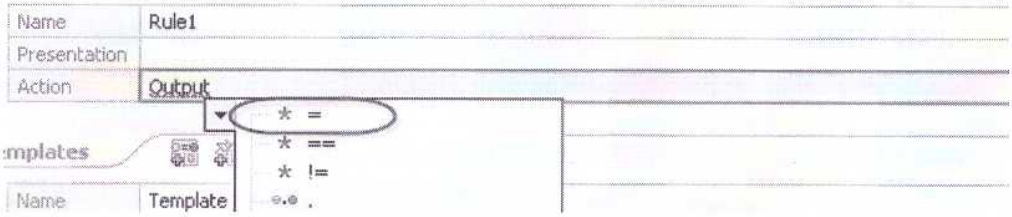
___p. Click on the **Action** link in the **Action** field.

In the previous steps, you had ignored the popup list which appears when a field link is clicked to keep the steps simple. However, the popup lists can also be very useful in supplying the proper values for the fields. Let's try it in the next step.

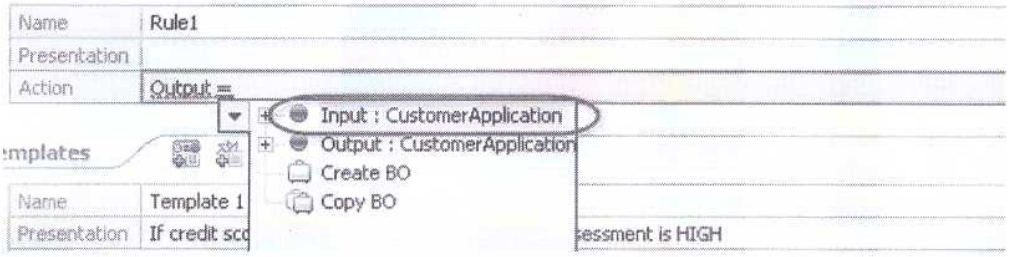
___q. From the popup list, select **Output : CustomerApplication**. (**Output** is a property from the Rule Set Interface, and **CustomerApplication** is the property type.)



___r. From the popup list, select the '=' operation.



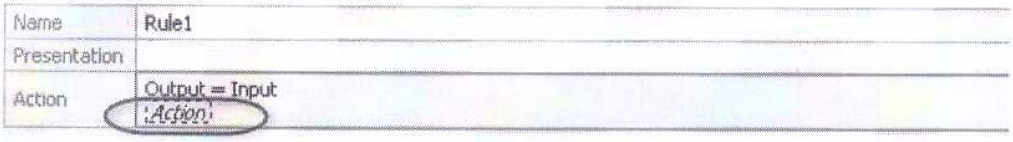
___s. From the popup list, select **Input : CustomerApplication**.



Setting up this **Action** in the Business Rule will copy the contents of the **Input** variable to the **Output** variable. The **Input** variable is a parameter passed into this Rule Set when this Business Rule is invoked. The **Output** variable will be passed back to the component invoking this Business Rule when execution is complete.

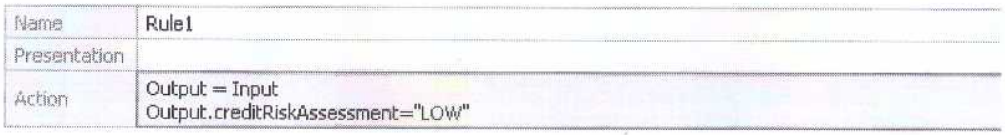
___t. While the cursor is still in the **Action** field, press the **Enter** key. This will add another **Action** link to the **Action** field.

___u. Click on the new **Action** link.

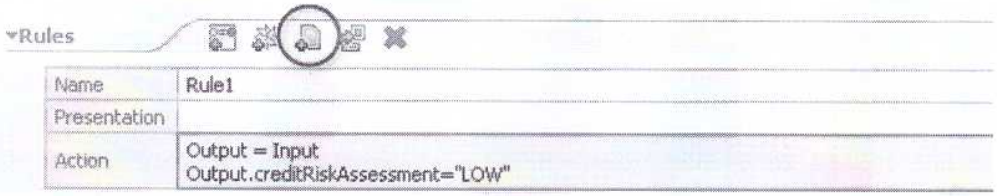


___v. Ignore the popup list again and type the following into the second line of the **Action** field:

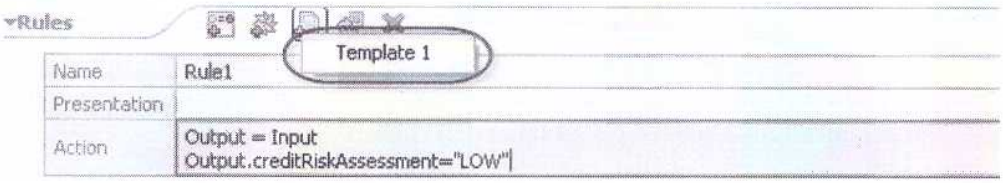
Output.creditRiskAssessment="LOW"



___w. Click on the **Add Template Rule** icon in the **Rules** section.

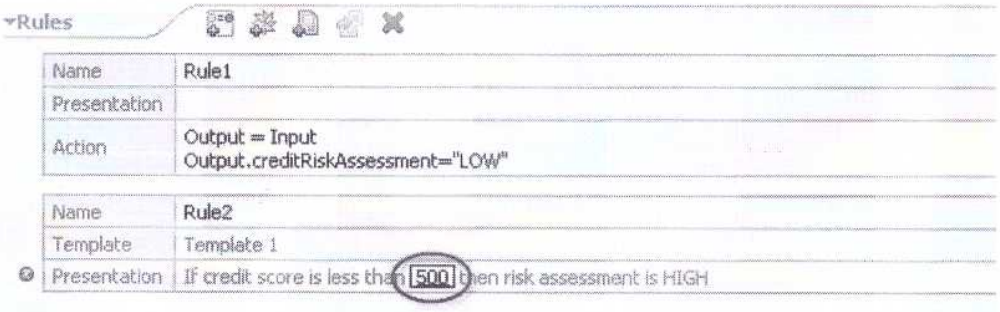


___x. From the popup list, select **Template 1**.



This will create an empty Rule based on the Rule Template created earlier.

___y. Specify a value of **500** for the **threshold** variable in the **Presentation** field of **Rule2**.



The Business Rule is now complete. This Rule Set basically recreates the logic in the previous

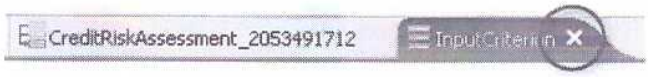
Java implementation where the risk assessment will be HIGH if the credit score is less than the current threshold level of 500. The significant difference is that this was done without any programming, and the threshold value can be changed dynamically at runtime. This provides tremendous flexibility in the execution of business processes.

____ 3. Save and Verify the Business Rule

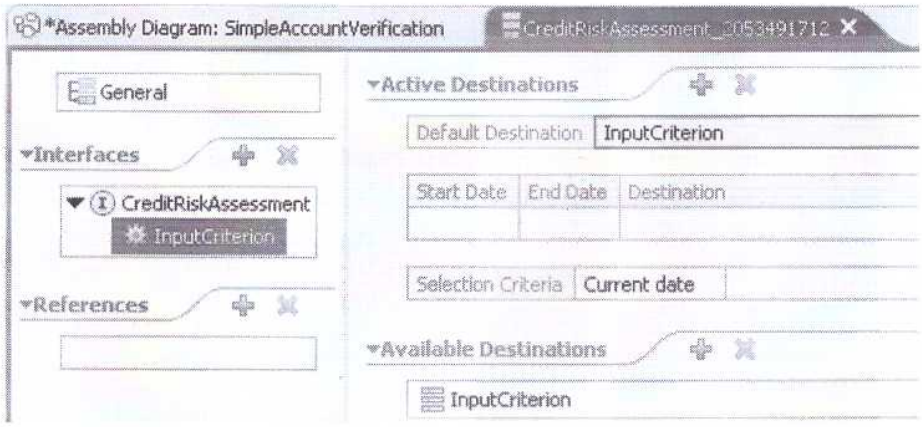
___a. Press **Ctrl-s** to save the **Rule Set**.

Hint: A red 'x' mark ( ) might still be visible beside the **Presentation** field of **Rule2**. This should disappear when the parent Rule Group is also saved.

___b. Close the **InputCriterion** Rule Set editor.



Focus should return to the **CreditRiskAssement_2053491712** Business Rule Group.



___c. Press **Ctrl-s** to save the **Business Rule Group**.

___d. Close the **CreditRiskAssement_2053491712** Business Rule Group editor.

Focus should return to the **Assembly Diagram** editor.

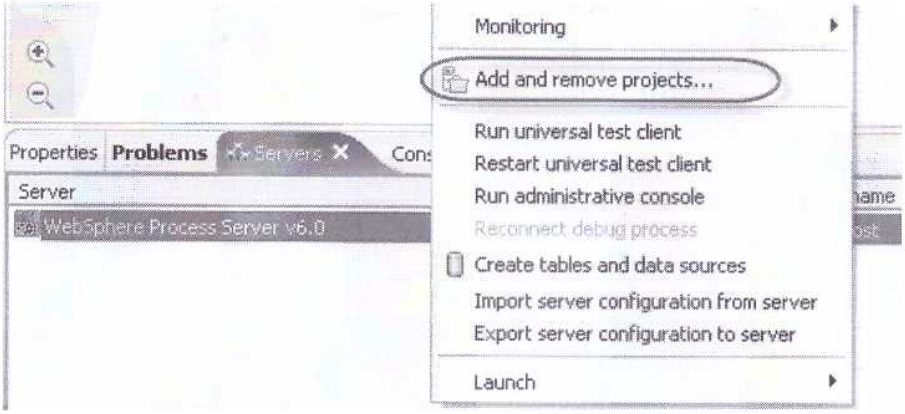
___e. From the **Assembly Diagram** editor, press **Ctrl-s** to save the modified assembly.

___f. Switch to the **Problems** view. Verify that no errors exist (messages with a red 'x' mark). Warnings and informational messages are acceptable.

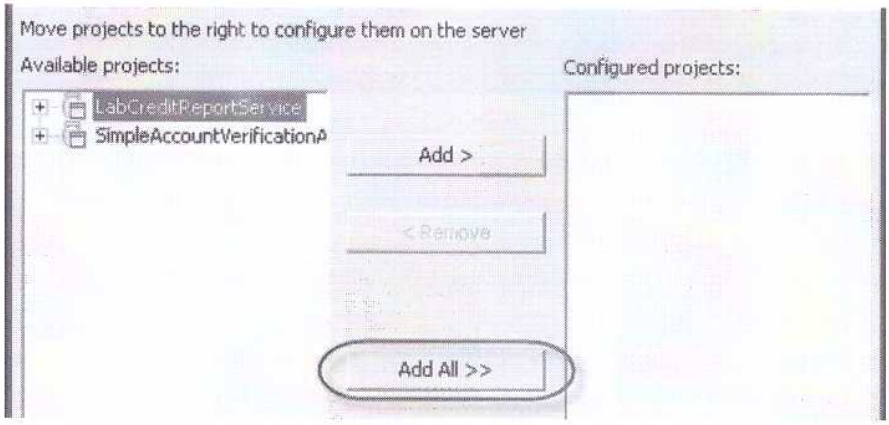
____ 4. Retest the SimpleAccountVerification Process

___a. Switch to the **Servers** view.

___b. Right-click on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Add and remove projects**.



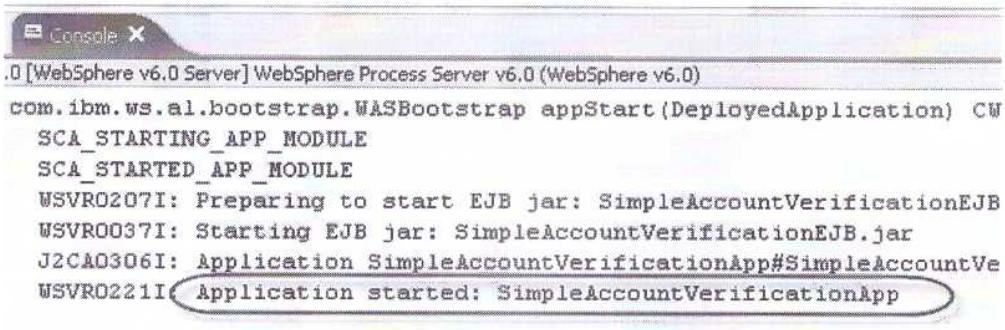
___c. From the **Add and Remove Projects** window, click on **Add All**.



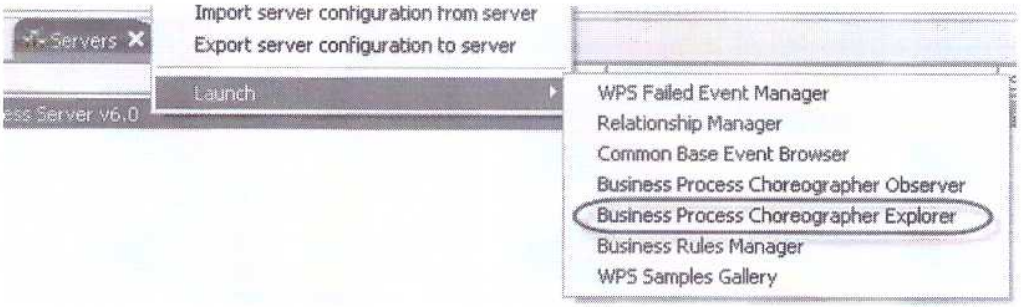
___d. Click on **Finish**. This will add the business process to the test server configuration.

___e. A progress indicator will appear in the lower-right corner of the WebSphere Integration Developer window. Wait for this indicator to disappear before proceeding to the next step.

___f. Focus will automatically switch to the **Console** view. Verify that a message appears in the **Console** view indicating that the **SimpleAccountVerificationApp** process has started. Scroll right as needed.



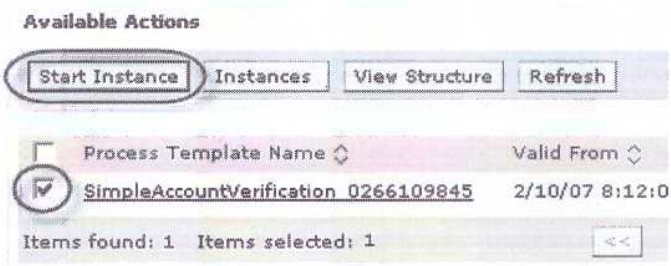
___g. Switch to the **Servers** view and right-click on **WebSphere Process Server v6.0**. From the popup menu, select **Launch->Business Process Choreographer Explorer**.



The Business Process Choreographer Explorer appears. This is a built-in web application that provides a way to manage business processes, as well as work items associated with human tasks in the process.

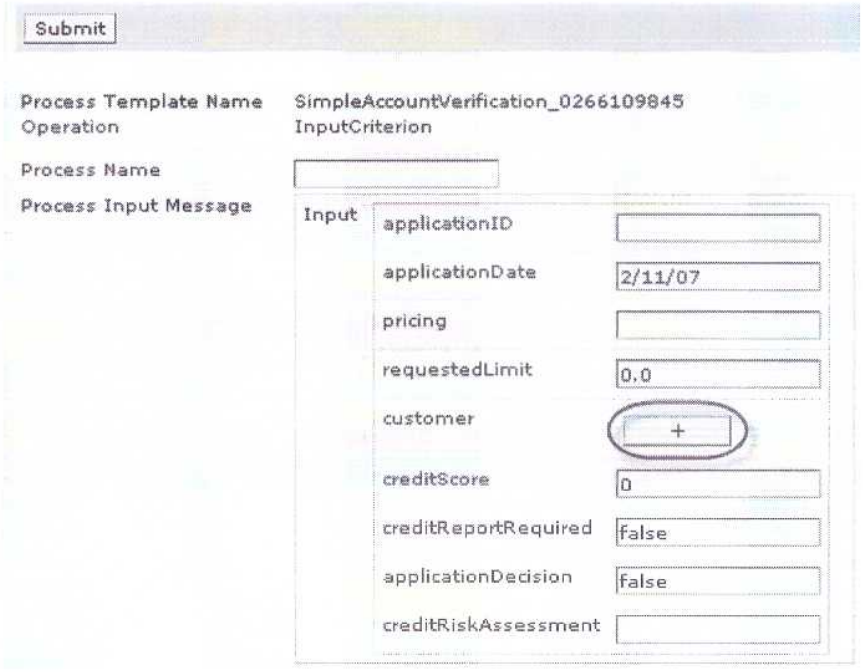
___h. Click on the **My Process Templates** link to display a list of processes which can be started.

___i. Place a checkmark beside the **SimpleAccountVerification** process template, then click on **Start Instance**.

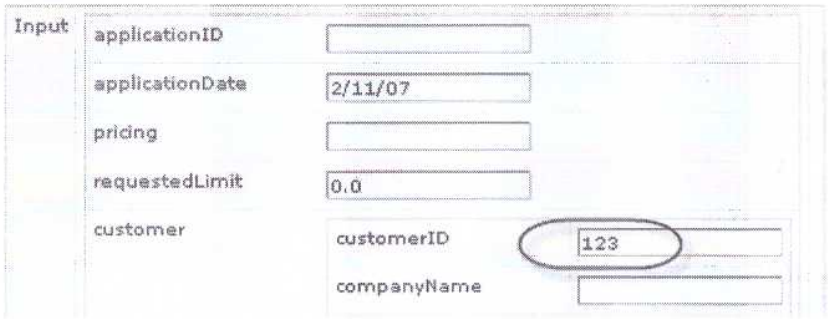


This will start the SimpleAccountVerification business process, and display the default input page.

___j. From the default input page, click on the Expand button.



___k. For the **customerID** field, type **123**, accept the defaults for the other fields, then click on **Submit**.



A Customer ID of "**123**" will again result in a low credit score and a denied request. In this case, it used the Business Rule to determine that the credit risk assessment is **LOW** because the credit score is below the current threshold of **500**. Based on the risk rating, the process automatically rejected the request.

```
O ********************
O * Call Web Service  *
O ********************
O >>>>>>>>>>>>>>>>>>>>
O >>>>> Credit Score = 449
I   J2CA0086W: Shareable conne
positoryDataSource was used with
O >>>>> Request is denied!.
```

However, because the credit risk assessment is now implemented as a Business Rule, the next few steps will illustrate how this will provide greater business flexibility.

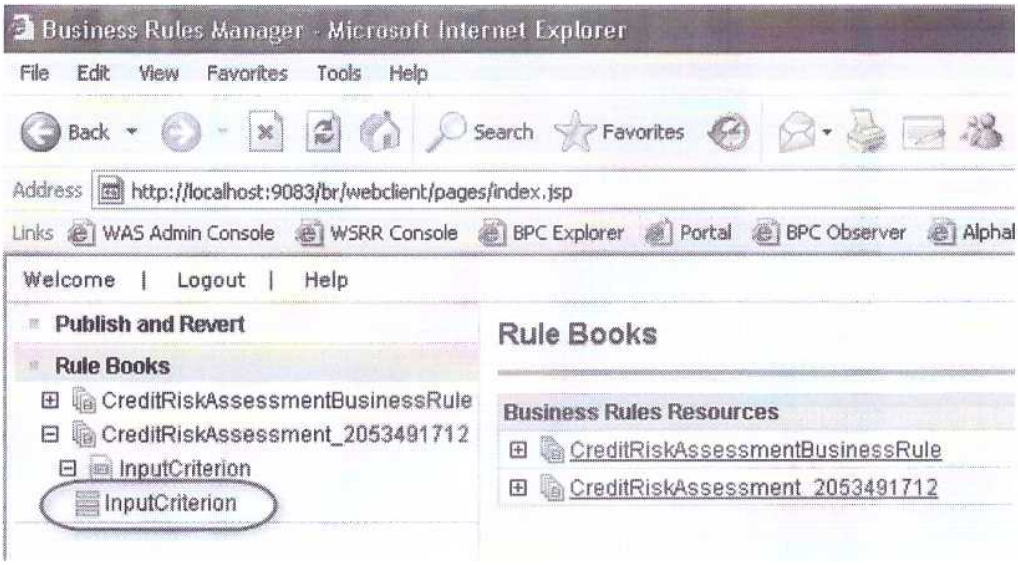
____ 5. Modify the Credit Assessment Business Rule at Runtime

___a. Click on the **Internet Explorer** icon in the **Quick Launch** bar. There is also an **Internet Explorer** icon on the desktop.

___b. From the **Internet Explorer** window, click on the **Business Rules Manager** link. The link will open the URL http://localhost:9083/br/webclient/ .

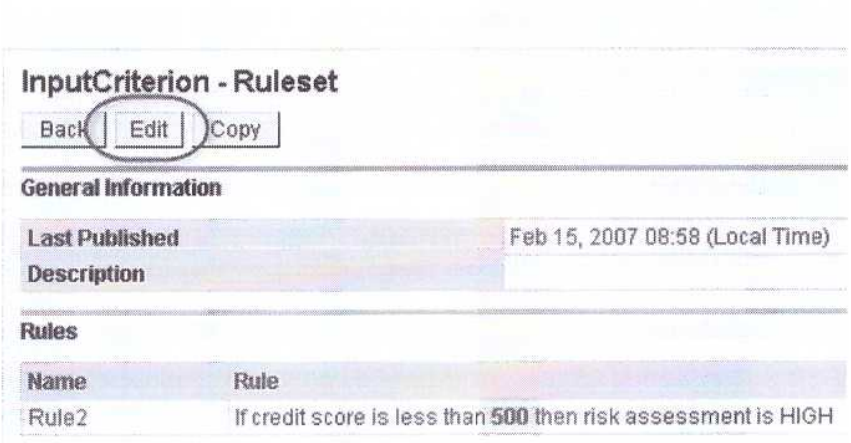
Using a standard web browser, an authorized person such as a manager or business analyst, can dynamically change the behavior of the business process at runtime by modifying the externalized business rules.

___c. From the **Rule Books** tree view on the left, click on **InputCriterion**.



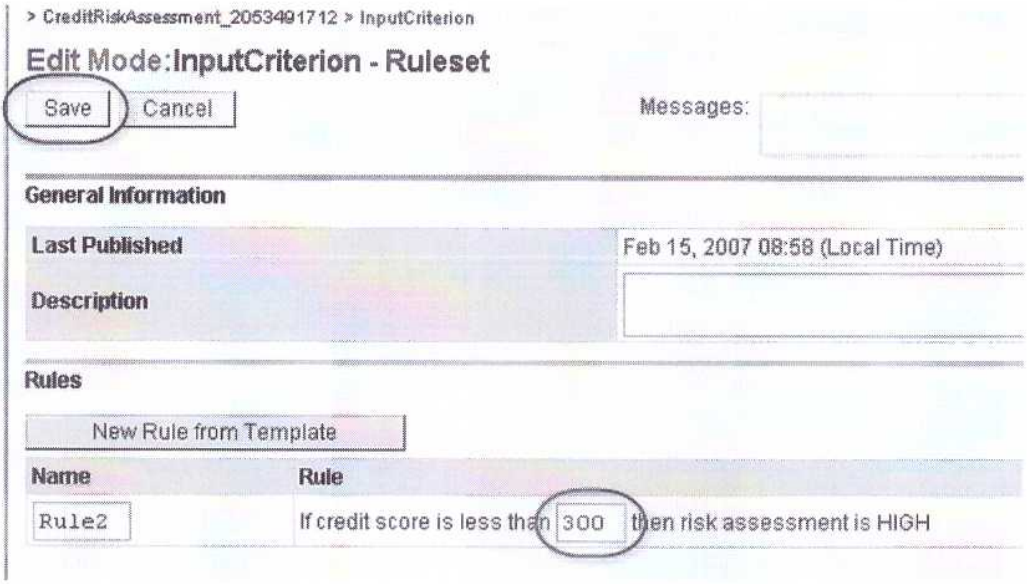
The logon window did not appear because security is disabled in the test server. In a secure production environment, users will be required to log in and business rules can only be accessed by authorized users.

___d. From the **InputCriterion Ruleset** pane, click on **Edit**.



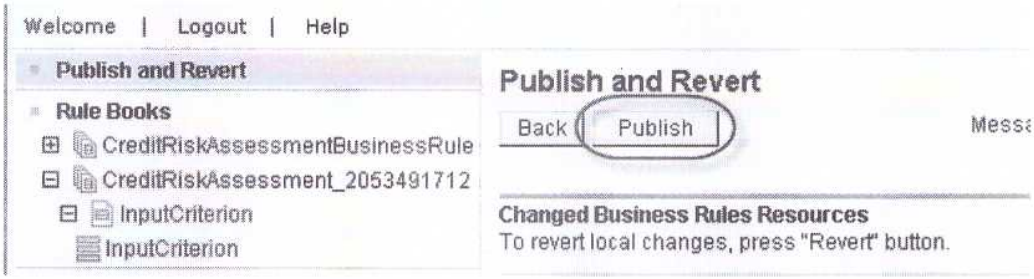
This is the ruleset for the Credit Risk Assessment Business Rule. The current threshold of 500 will be lowered so that the request of Customer ID **123** will be accepted.

___e. From the Ruleset editor, change the threshold value to **300**, and then click on **Save**.

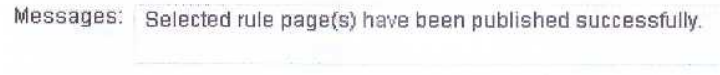


___f. Click on the **Publish and Revert** link.

___g. From the **Publish and Revert** pane, click on **Publish**.



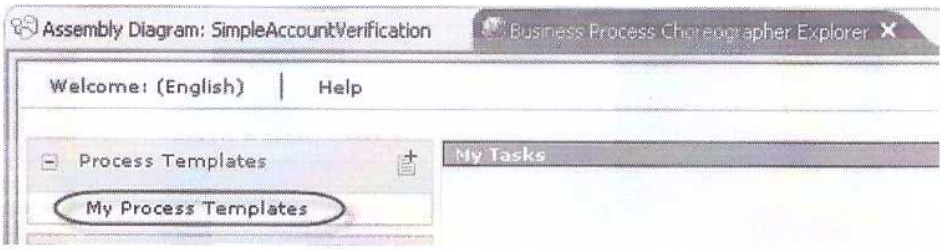
The **Messages** box should indicate that the rule change was published successfully.



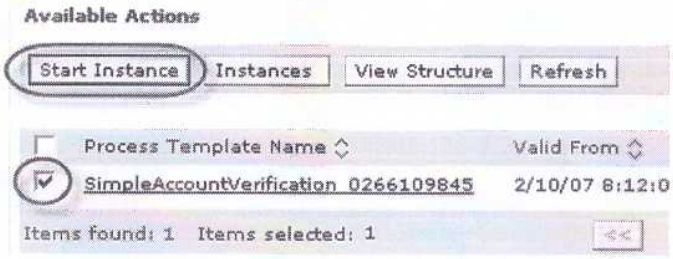
___h. Minimize the **Internet Explorer**.

___i. Switch to the **WebSphere Integration Developer**.

___l. From the Business Process Choreographer Explorer, click on the **My Process Templates** link to display a list of processes which can be started.

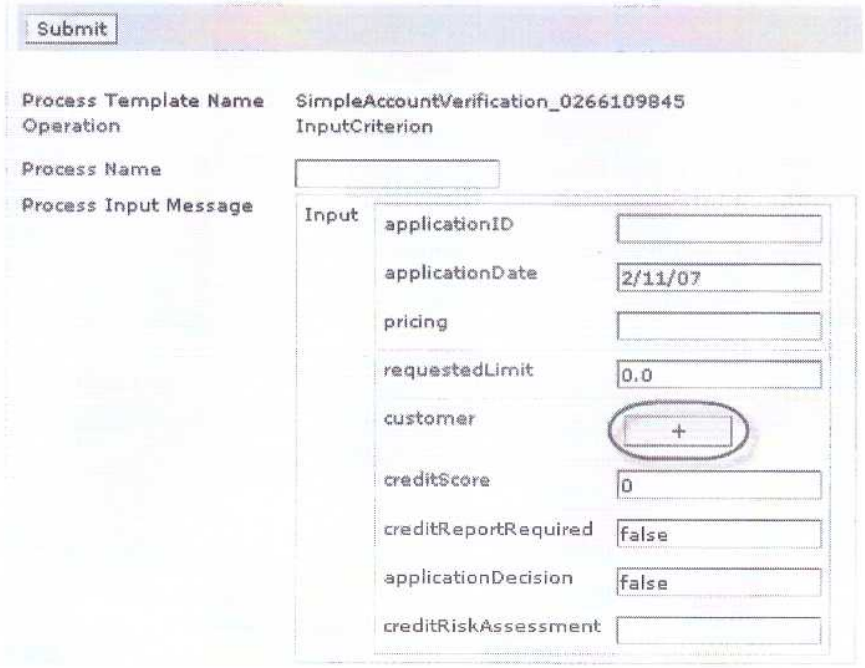


___m. Place a checkmark beside the **SimpleAccountVerification** process template, then click on **Start Instance**.

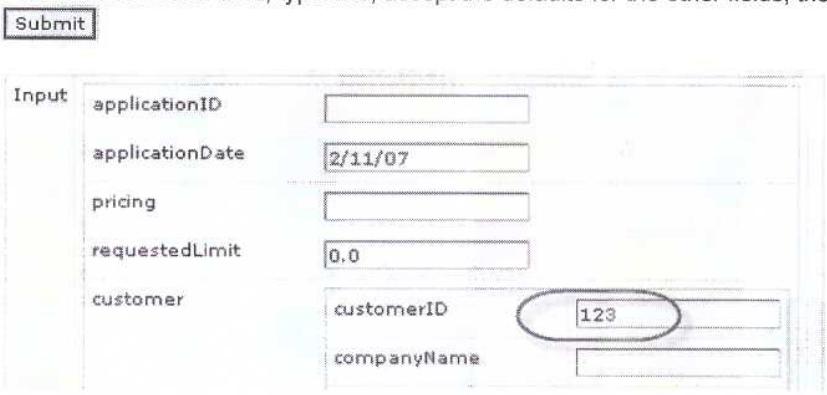


This will start the SimpleAccountVerification business process, and display the default input page.

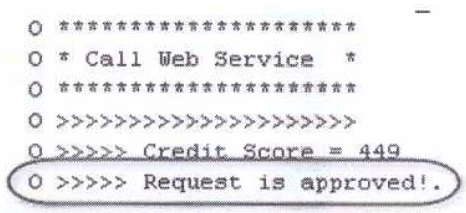
___n. From the default input page, click on the Expand button.



___j. For the **customerID** field, type **123**, accept the defaults for the other fields, then click on **Submit**.



A Customer ID of "**123**" should now result in an accepted request because the threshold level in the Credit Risk Assessment Business Rule was lowered to 300.



Significant cost reductions can result from this type of business and IT flexibility.

____ 6. Reset the Credit Risk Assessment Business Rule

___a. When you're done testing the business process, switch to the **Servers** view.

___b. Right-click on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Add and remove projects**.

___c. From the **Add and Remove Projects** window, click on **Remove All**.

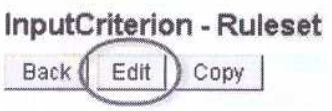
___d. Click on **Finish**.

___e. Close the **Business Process Choreographer Explorer.**

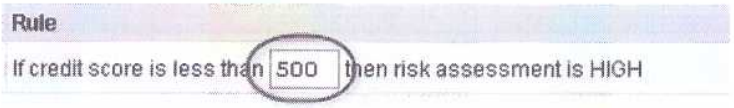
___f. Switch to the **Internet Explorer** with the Business Rules Manager page.

___g. Click on the **InputCriterion** link again.

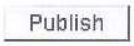
___h. Click on **Edit**.

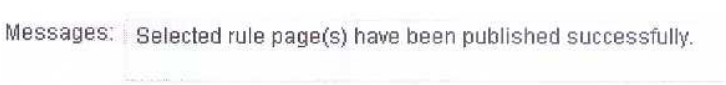


___i. Change the threshold value back to **500**, then click on **Save**.



___j. Click on the **Publish and Revert** link. 

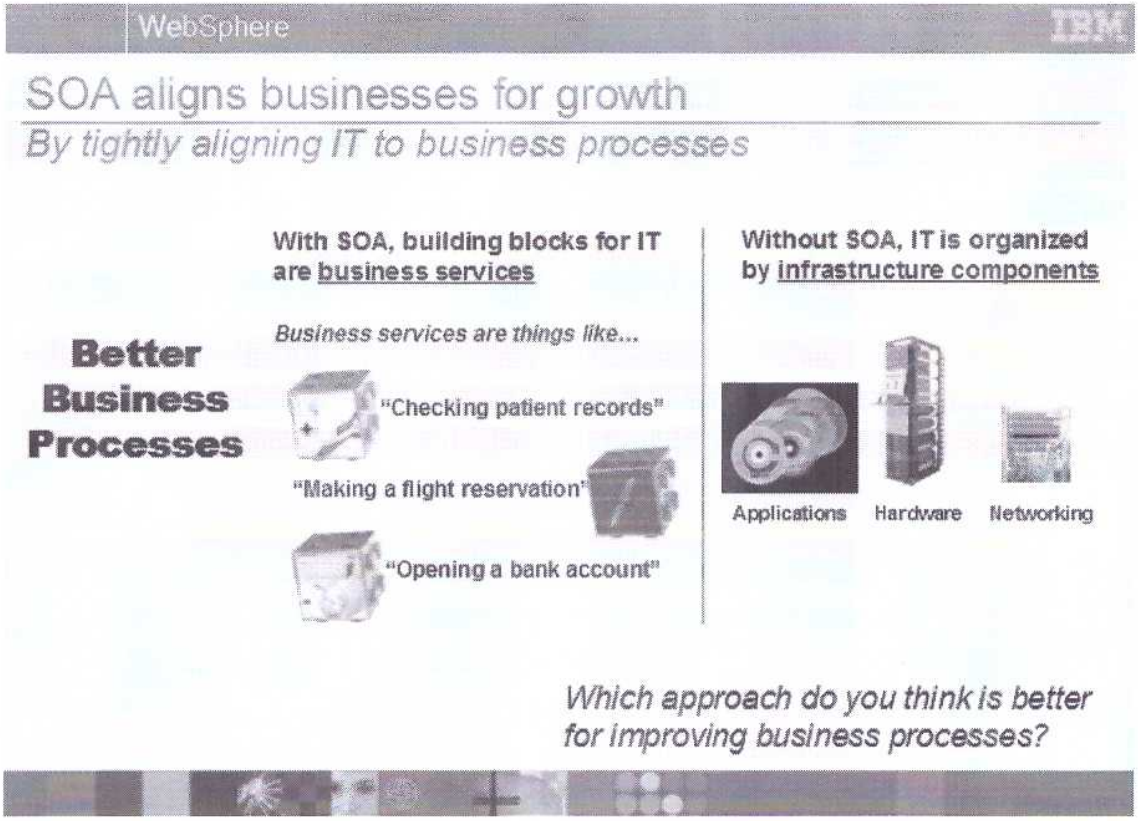
___k. Click on the **Publish** button. 



___l. Close the **Internet Explorer**.

Please wait for the next lecture before proceeding with the lab.

Part 7: Human Tasks

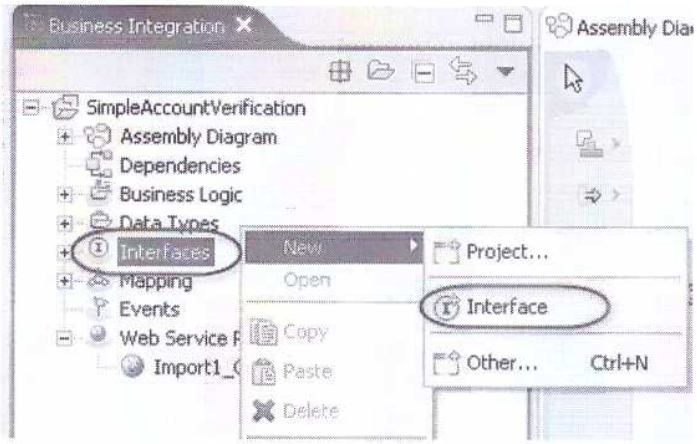


Business processes often involve human intervention, such as a loan officer reviewing loan applications, or a manager approving employee reimbursements. There will always be situations where systems or programs cannot automatically make decisions based on the available information. A business process might also involve a trivial task such as a secretary manually sending fax requests to print promotion materials. Because the goal is to implement the entire process end-to-end, then manual tasks need to be included with the automated tasks. With IBM's SOA Foundation and the WebSphere Process Integration software portfolio, human activities are fully supported in business processes. In fact, the human task capabilities in the WebSphere Process Server are also designed for loose-coupling, effective reuse, and the building-block approach.

In this section, the implementation type for the **Pricing and Approval** task will be converted from Java to a human task. The **SimpleAccountVerification** BPEL process will also be modified to add a new task called **Generate Acceptance**. The **Generate Acceptance** task is being added simply to display output from the **Pricing and Approval** human task. This will also help introduce BPEL development.

____ 1. Define the Interface for the new 'Generate Acceptance' task.

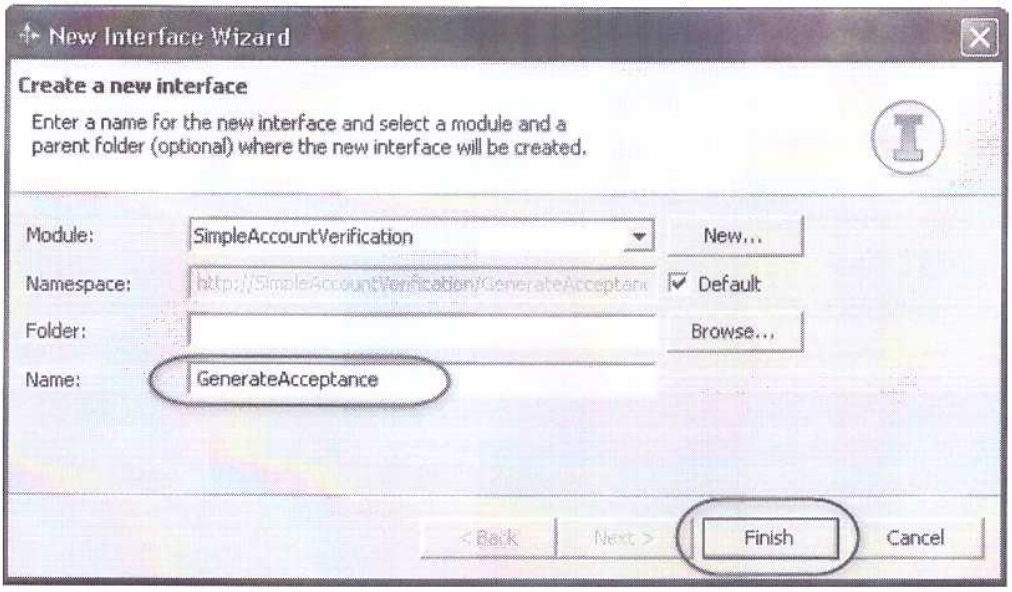
___a. From the **Business Integration** tree view, right-click on **Interfaces**. From the popup menu, select **New -> Interface**.



An interface first needs to be defined for the Human Task. An interface consists of operations and the inputs and outputs associated with each operation. A Human Task component is handled just like any other service component, so it also needs to be invoked using a clearly defined set of operations and its associated inputs and outputs.

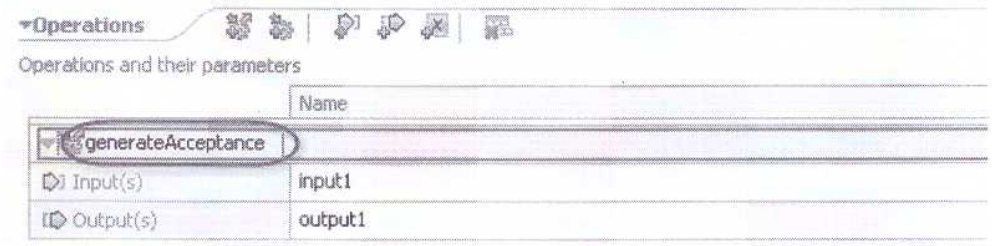
At first it might be a scary thought knowing that we humans can now be invoked just like any other software component, but this is actually a powerful feature. Because the Human Task component is also a generic building block where the internals are hidden and only the interface is exposed, then it becomes easy to swap different building blocks even though the internals involve different technologies, programming languages, and operating environments. This is one of the ways IT flexibility is achieved.

___b. From the **New Interface Wizard** window, specify **GenerateAcceptance** for the **Name**, then click on **Finish**.

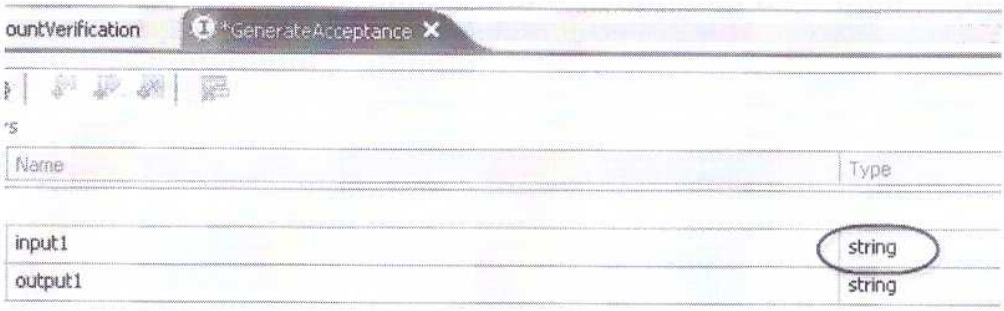


___c. From the **GenerateAcceptance** Interface editor, click on the **Add Request Response Operation** icon.

___d. Specify **generateAcceptance** as the **operation** name.

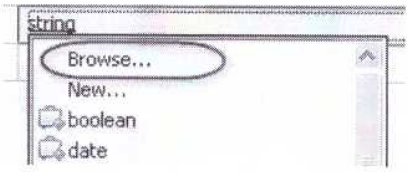


___e. Click on the **string** link for **input1**.

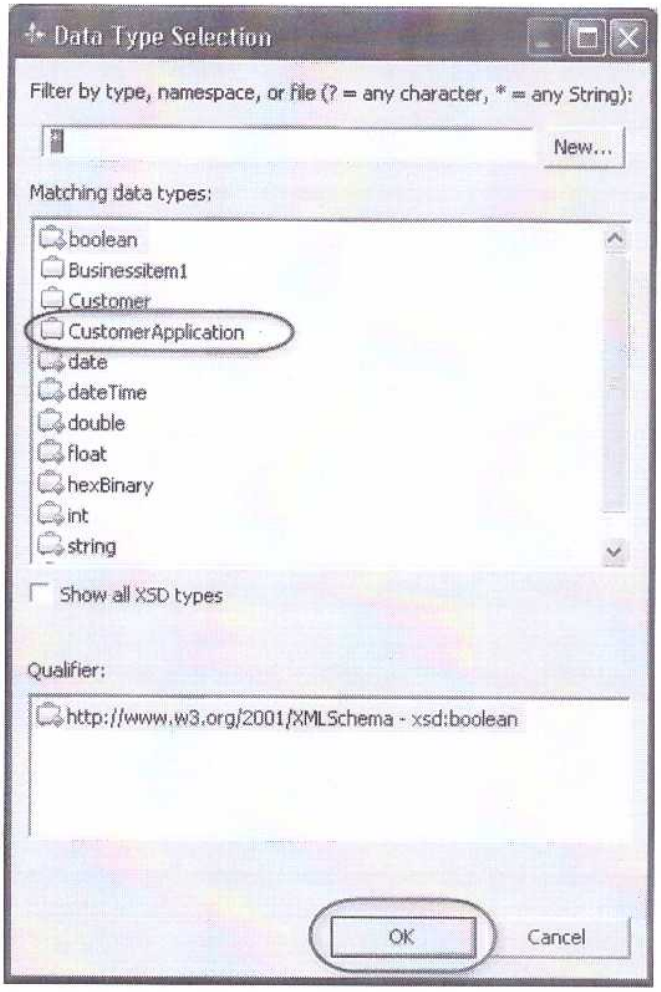


When the **generateAcceptance** operation is invoked, a parameter variable named **input1** will be passed. By default, the parameter **input1** is defined as a string type. This will need to be changed to use the **CustomerApplication** business object type instead.

___f. From the popup list, select **Browse**.



___g. From the **Data Type Selection** window, select **CustomerApplication**, then click on **OK**.



___h. Change also the type of **output1** to **CustomerApplication**.

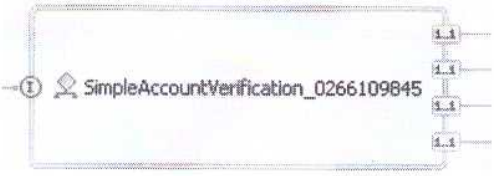
During execution, when the **generateAcceptance** operation has completed, a result variable named **output1** will be passed back to the calling component.

___i. Press **Ctrl-s** to save the new Interface.

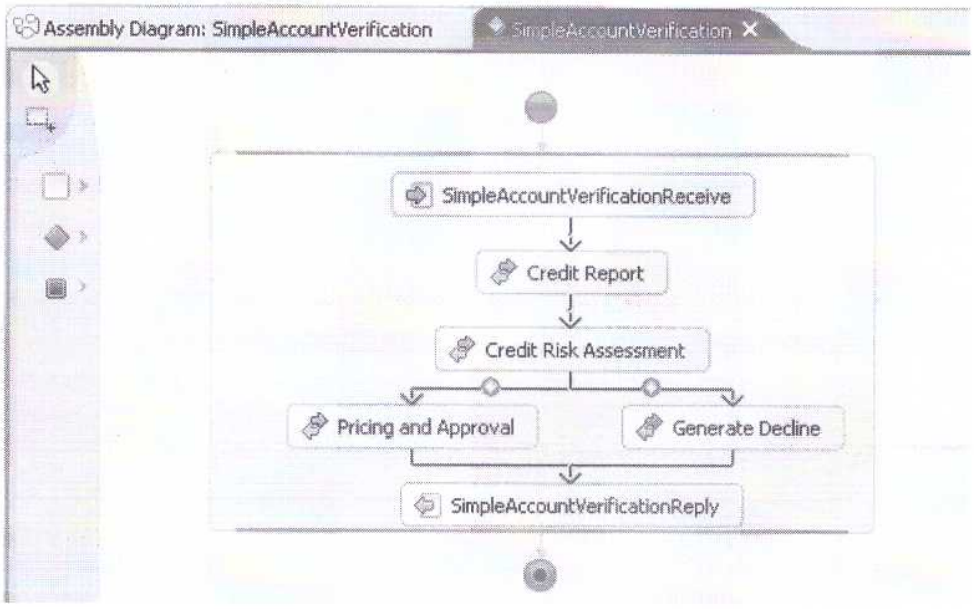
___j. Close the **GenerateAcceptance** Interface editor.

____ 2. Add the GenerateAcceptance Task to the SimpleAccountVerification BPEL Process

___a. From the **Assembly Diagram editor**, double-click on the **SimpleAccountVerification** component.



The **SimpleAccountVerification** BPEL editor will appear.

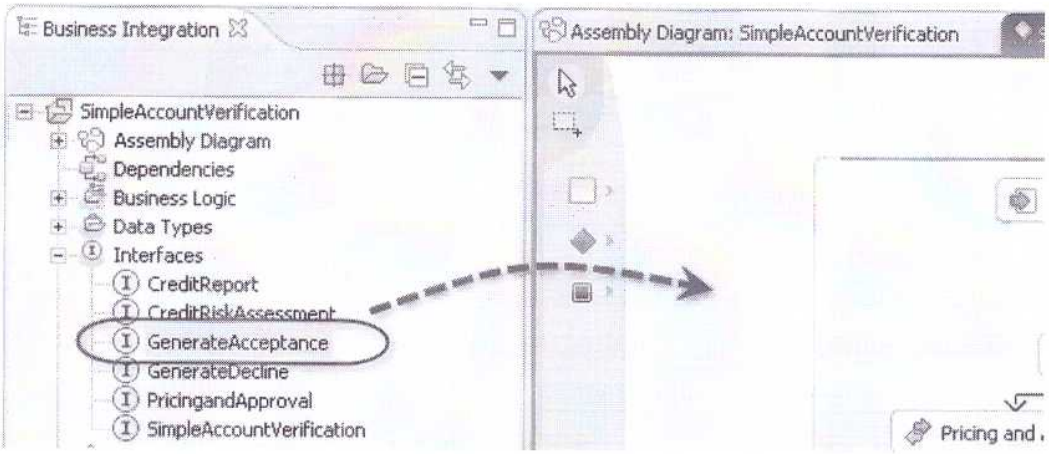


This is the BPEL version of the **SimpleAccountVerification** business model developed earlier using the WebSphere Business Modeler.

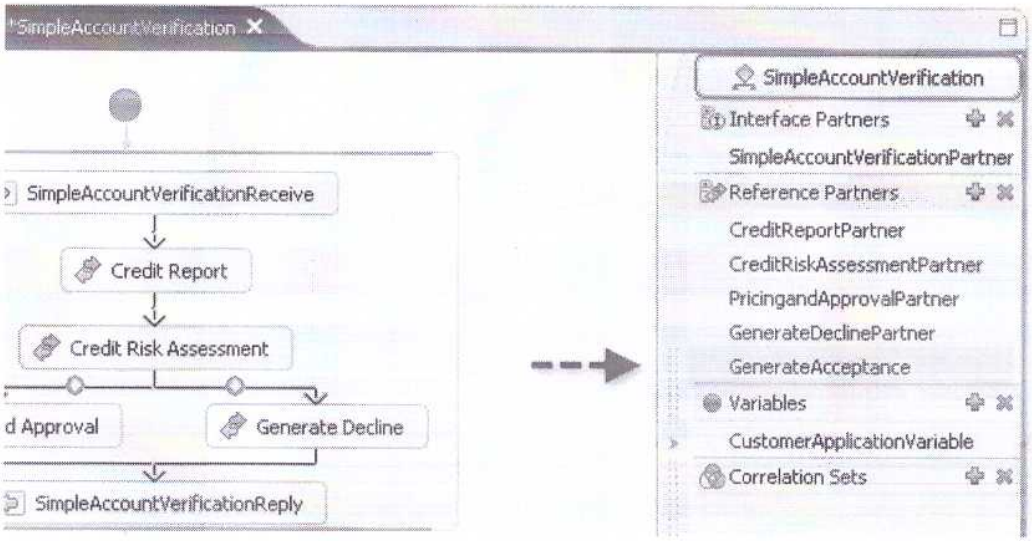
___b. Right-click on an empty space in the BPEL editor. From the popup menu, select **Align Parallel Activities Contents Automatically**.

This option will make the editor always automatically adjust the BPEL diagram to the optimum layout.

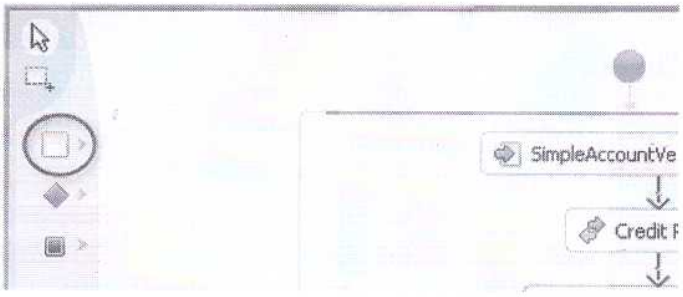
___c. From the tree view and using the left mouse button, drag and drop the **GenerateAcceptance** Interface to an empty space in the BPEL editor.



This will add the **GenerateAcceptance** Interface to the list of Reference Partners, making it available for use by any task or activity in the BPEL process.

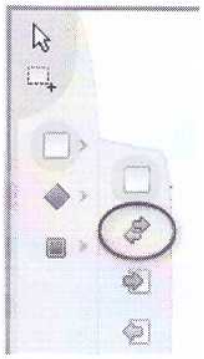


___d. From the **Activity Palette** of the BPEL editor, click on the **Empty Action** icon.



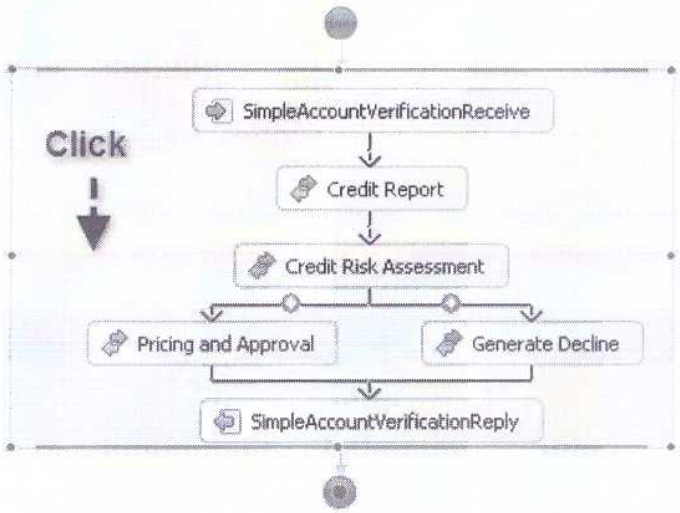
This will open a sub-palette.

___e. From the sub-palette, click on the **Invoke** icon.



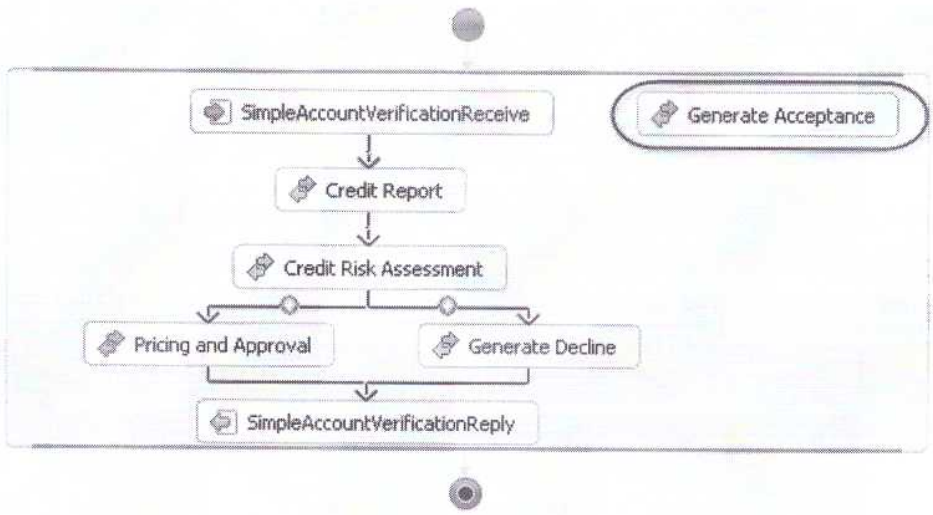
This will load the cursor with the **Invoke** activity and is ready to be dropped on to the BPEL diagram.

___f. Click on an empty space in the BPEL diagram.

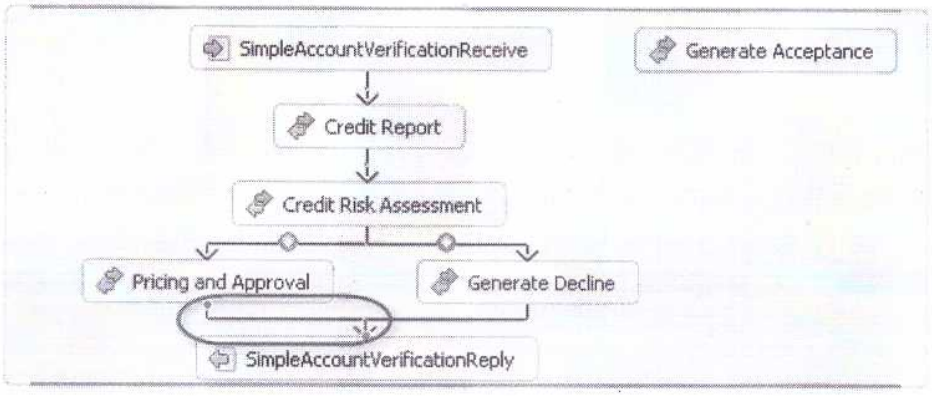


This will add a new Invoke activity to the BPEL diagram.

___g. Change the **name** of the new Invoke activity to **Generate Acceptance**.

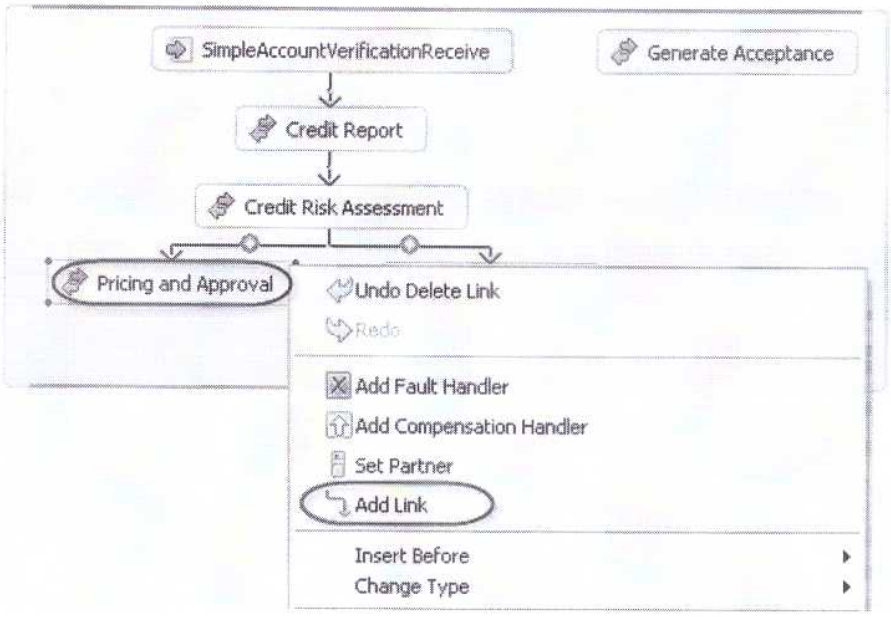


___h. Click to select the link between the **Pricing and Approval** activity and the **SimpleAccountVerificationReply** activity.



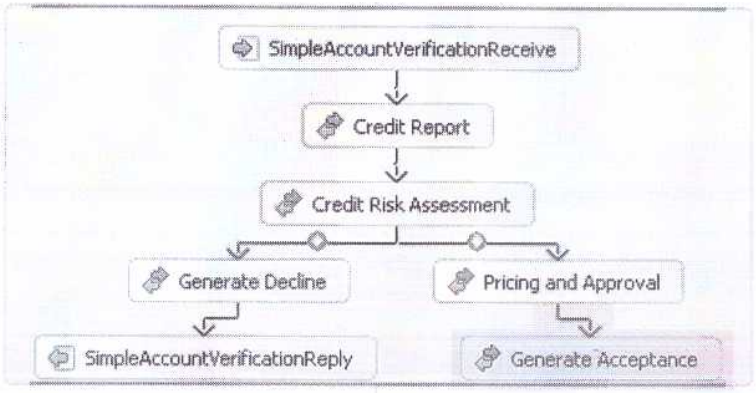
___i. Press the **Delete** key.

___j. Right-click on the **Pricing and Approval** activity. From the popup menu, select **Add Link**.



This will start a connection.

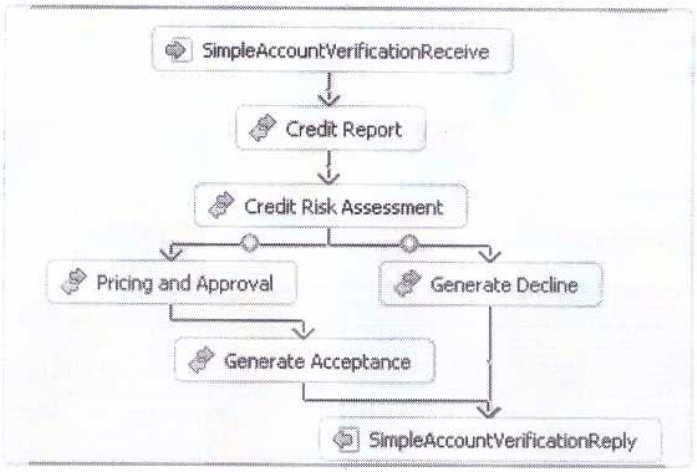
___k. Click on the **Generate Acceptance** activity to complete the connection.



___l. Right-click on the **Generate Acceptance** activity. From the popup menu, select **Add Link**.

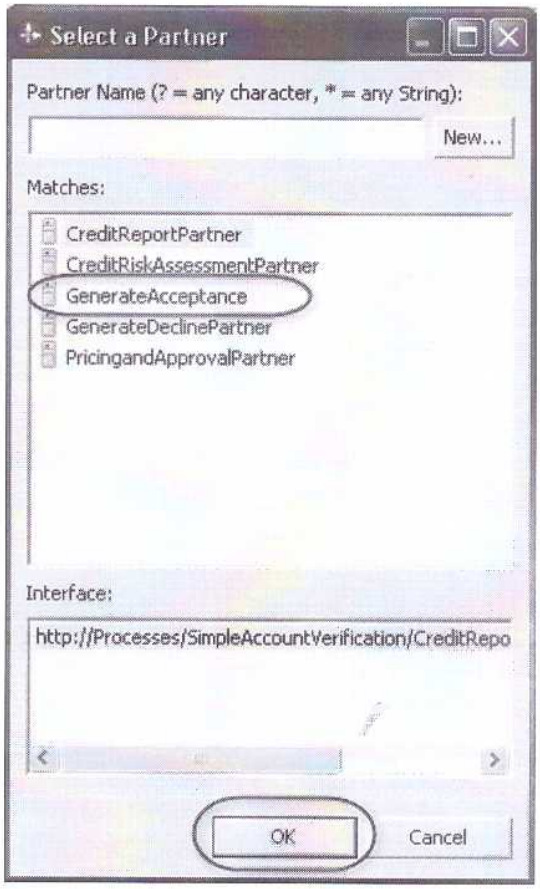
___m. Click on the **SimpleAccountVerificationReply** activity to complete the connection.

___n. Verify that the SimpleAccountVerification BPEL process looks similar to the screenshot below.

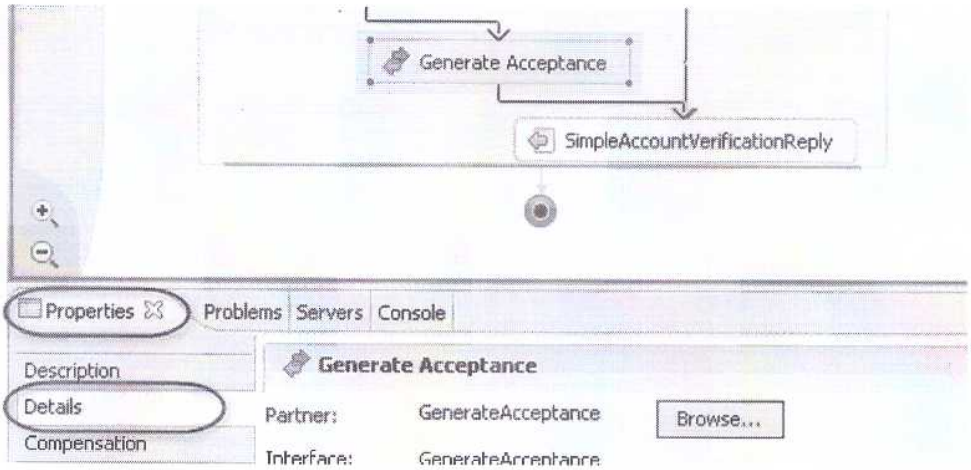


___o. Right-click on the **Generate Acceptance** activity. From the popup menu, select **Set Partner**.

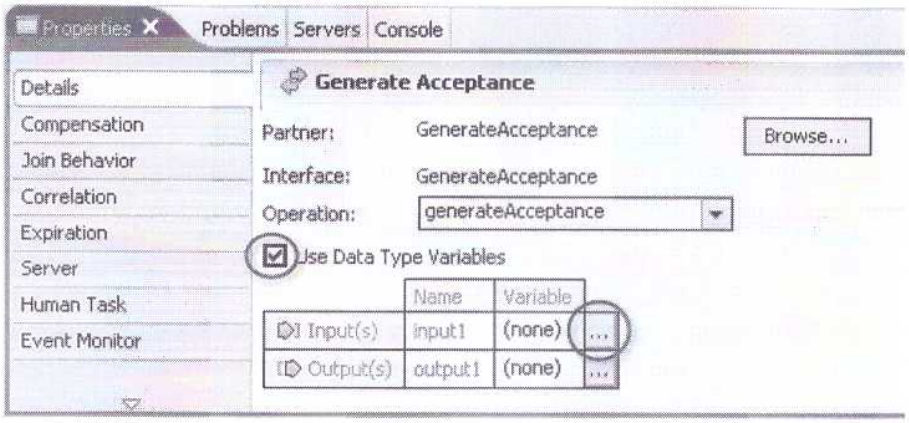
___p. From the Select a Partner window, select **GenerateAcceptance**, then click on **OK**.



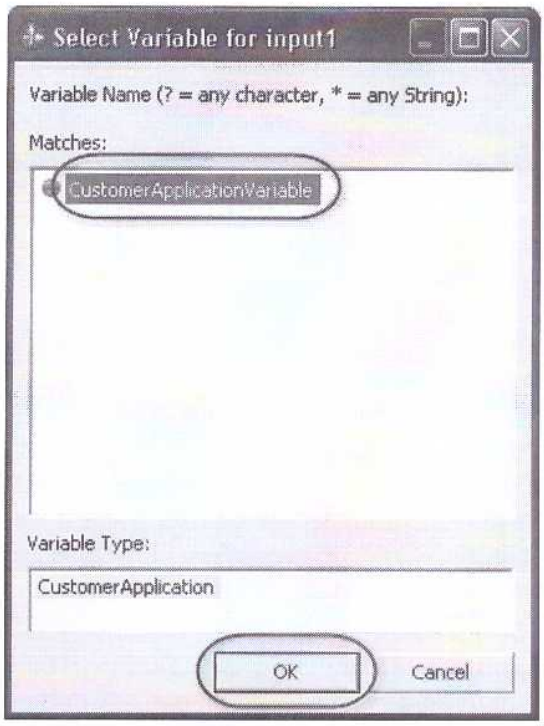
___q. With the **Generate Acceptance** activity still selected, switch to the **Properties** view and click on the **Details** tab.



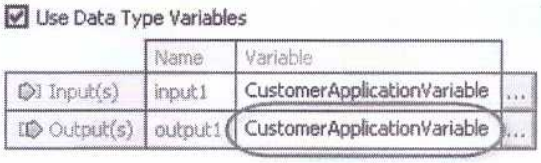
___r. Select the **Use Data Type Variables** option, then click on the **More** (...) button for the **Inputs** field.



___s. From the **Select Variable for input1** window, ensure that **CustomerApplicationVariable** is selected, then click on **OK**.



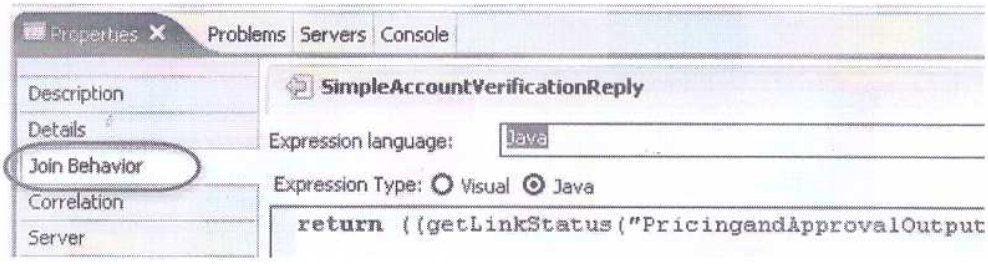
___t. Perform the same steps to also specify **CustomerApplicationVariable** for the **Output**.



These configuration changes specify that the **CustomerApplication** business object will be used for both the input and output of the **GenerateAcceptance** activity.

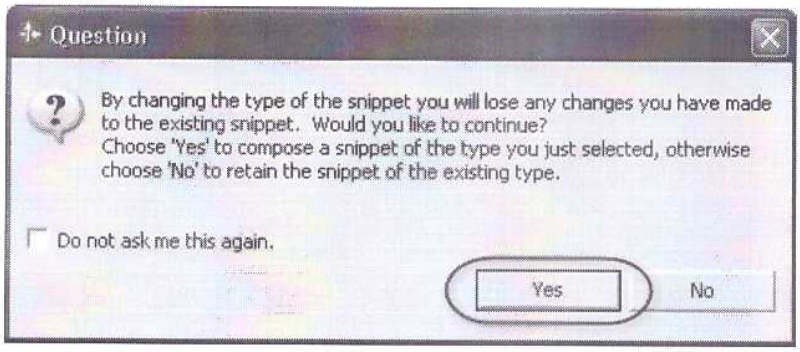
___u. Click on the **SimpleAccountVerificationReply** activity. 

___v. From the **Properties** view, select the **Join** tab.

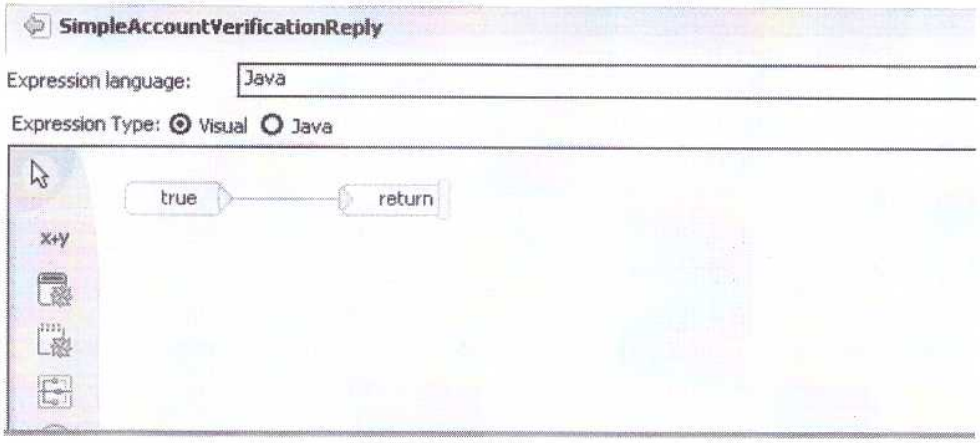


___w. Select the **Visual** option for the **Expression Type**.

___x. From the **Question** window, click on **Yes**.



The **Visual Snippet Editor** appears which will allow you to implement logic or expressions without having to know Java programming.



The default expression in the Visual Snippet Editor simply indicates that a boolean value of ***true*** will be returned to the caller of the **SimpleAccountVerification** process. At this point, no changes are necessary.

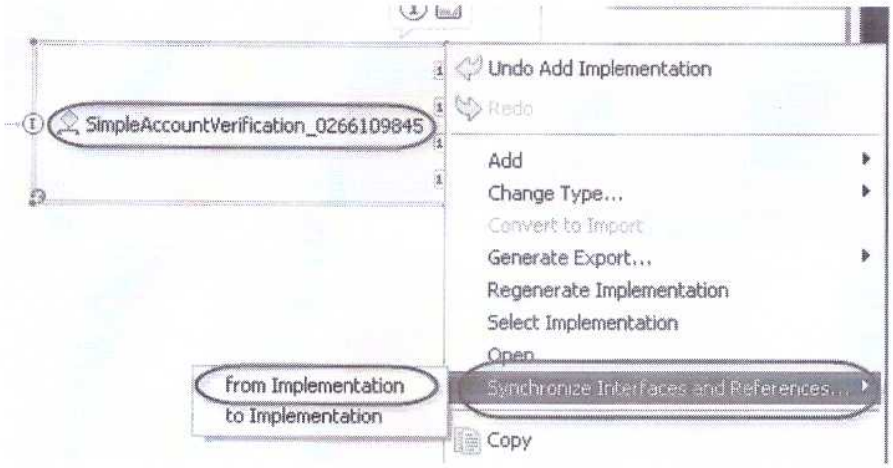
___y. Press **Ctrl-s** to save the changes to the BPEL process.

___z. Close the **SimpleAccountVerification** BPEL process editor.

Notice that as you were developing the BPEL process, the actual implementation for the tasks or activities in the process were never specified. You only associated the activities to partners or references to external implementations. This provides the loose-coupling characteristic needed for more effective modularization. The actual implementations are defined externally using the Assembly Diagram editor. This approach also provides a level of abstraction to components such as the BPEL process for greater IT flexibility.

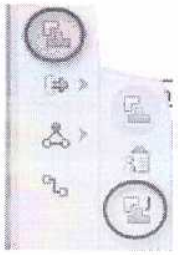
____ 3. Add the implementation for the new Generate Acceptance task

___a. From the **Assembly Diagram** editor, right-click on the **SimpleAccountVerification** component. From the popup menu, select **Synchronize Interfaces and References -> from Implementation**.



This will remove the red 'x' mark ( ) on the **SimpleAccountVerification** component, and will also add an additional reference point to the right of the component for the new **GenerateAcceptance** activity. The next step is to link that new reference point to another simple Java implementation.

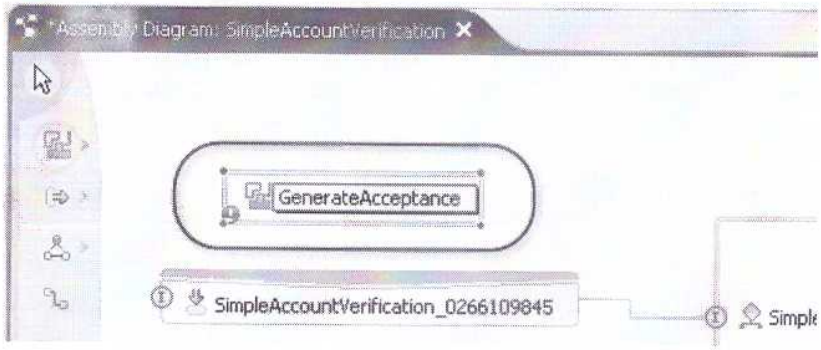
___b. From the palette on the left, click the top component to display a sub-palette. Select the **Java** component.



___c. Click on an empty space in the canvas to drop the **Java** component.

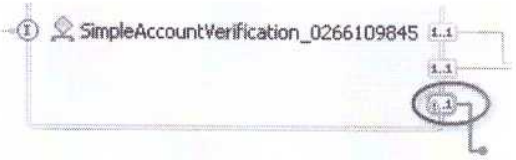
___d. Single-click on the new **Java** component to edit the name. Change the name to **GenerateAcceptance**.

Hint: You can also right-click on the new Java component, and from the popup menu, select **Rename**.



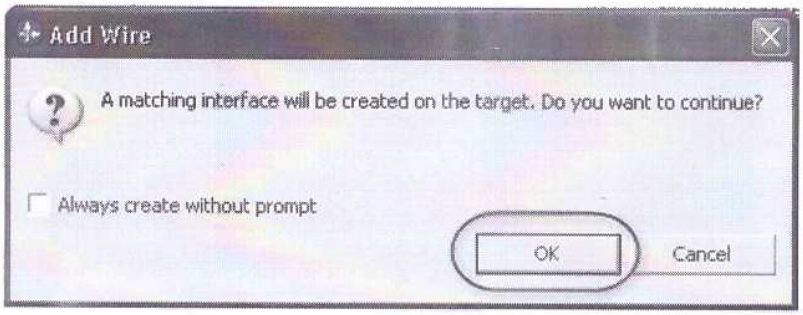
___e. From the palette, click on the **Wire** button.

___f. Click on the new **GenerateAcceptance** reference point on the **SimpleAccountVerification** process component. This will start a connection or wiring.



___g. Click on the **GenerateAcceptance** Java component. This will complete the new connection.

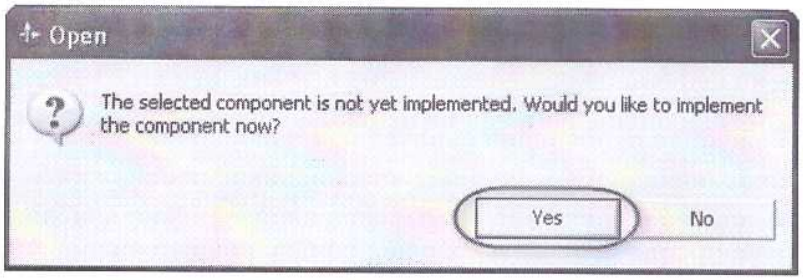
___h. From the **Add Wire** window, click on **OK**.



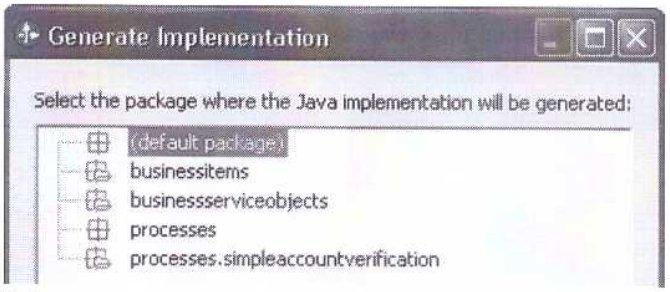
___i. From the palette, click on the **Selection Tool** button.

___j. Double-click on the **GenerateAcceptance** Java component.

___k. From the **Open** confirmation window, click on **Yes**.

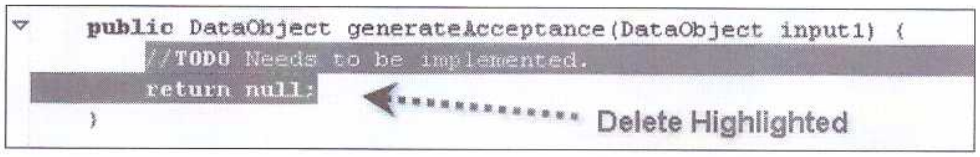


___l. From the **Generate Implementation** window, accept the default selection, then click on **OK**.

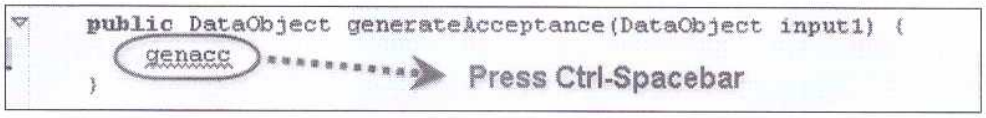


The Java editor appears for the implementation of the **GenerateAcceptance** component.

___m. Scroll down to the bottom of the Java code. Select the highlighted text below and **delete**.



___n. Type **credrep** in the line of code as shown below. Press **Ctrl-Spacebar**.



The **Ctrl-Spacebar** key combination activates a special feature called Code-Assist. This will add the required code snippet for you.

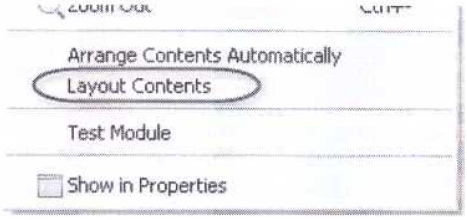
```
public DataObject generateAcceptance(DataObject input1) {
    /**
     *  CODE STUB
     *  Placeholder for production code later
     *  Code only for initial development and testing
     */
    System.out.println(">>>>>>>>>>>>>>>>>>>>>>>>>>>>");
    System.out.println(">>>>> Request is approved!.");
    System.out.println(">>>>> Pricing Plan has been set to "
            + input1.getString("pricing"));
    System.out.println(">>>>>>>>>>>>>>>>>>>>>>>>>>>>");
    return input1;
}
```

This code logic will display the Pricing Plan output from the **Pricing and Approval** human task. The **Pricing and Approval** human task will be implemented in the next steps.

___o. Press **Ctrl-s** to save the Java code, then close the Java editor.

____ 2. Convert the Pricing and Approval Java component to a Human Task

___a. From the **SimpleAccountVerification** assembly diagram editor, right-click on an empty space in the canvas. From the popup menu, select **Layout Contents**.



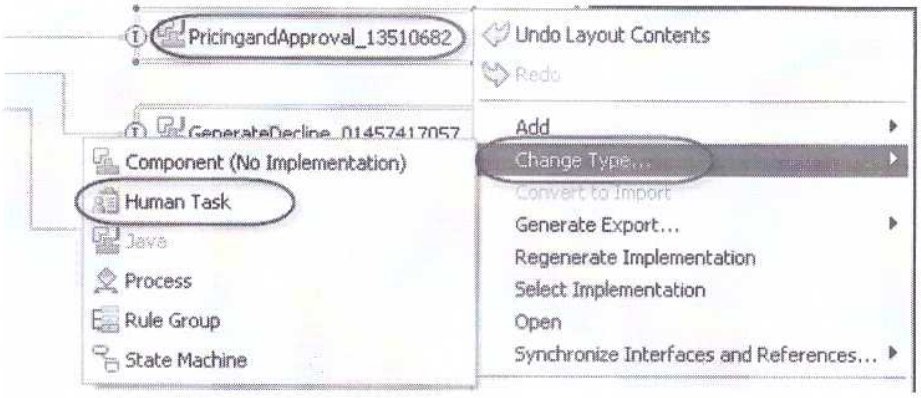
This will rearrange the components for a better layout.

___b. Press **Ctrl-s** to save the assembly diagram.

___c. If a progress indicator appears at the lower-right part of the main window, wait until it disappears indicating that the automatic build step is complete.

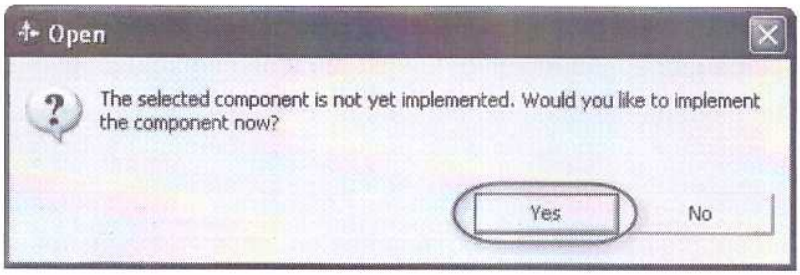
___d. From the **Assembly Diagram** editor, right-click on the **PricingandApproval** component.

___e. From the popup menu, select **Change Type -> Human Task**.

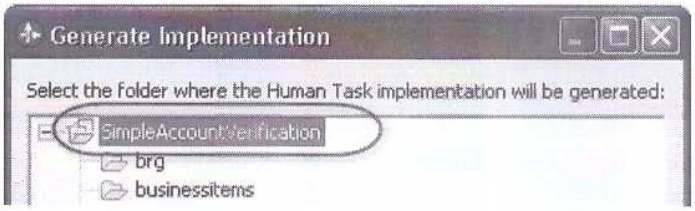


___f. Double-click on the **PricingandApproval** component, which is now a Human Task.

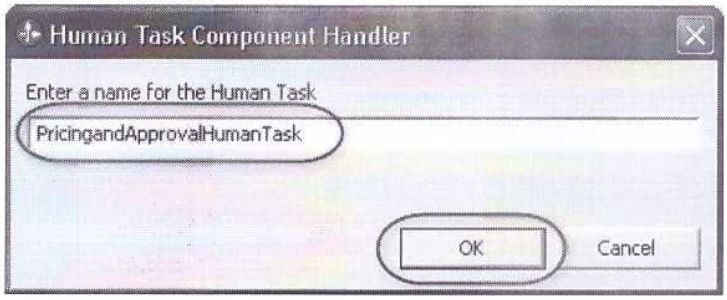
___g. From the **Open** confirmation window, click on **Yes**.



___h. From the **Generate Implementation** window, ensure that the **SimpleAccountVerification** folder is selected, then click on **OK**.



___i. From the **Human Task Component Handler** window, change the name to **PricingandApprovalHumanTask**, then click on **OK**.



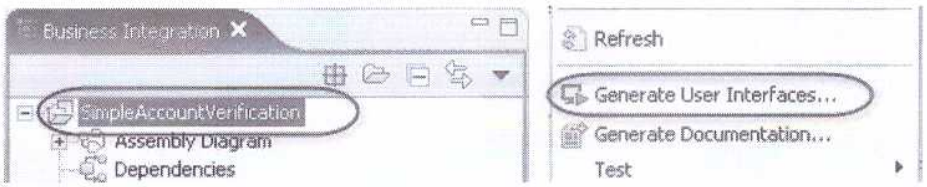
The Human Task editor will appear.

___j. Close the **Human Task** editor. At this point, we do not need to make any changes to the Human Task configuration.

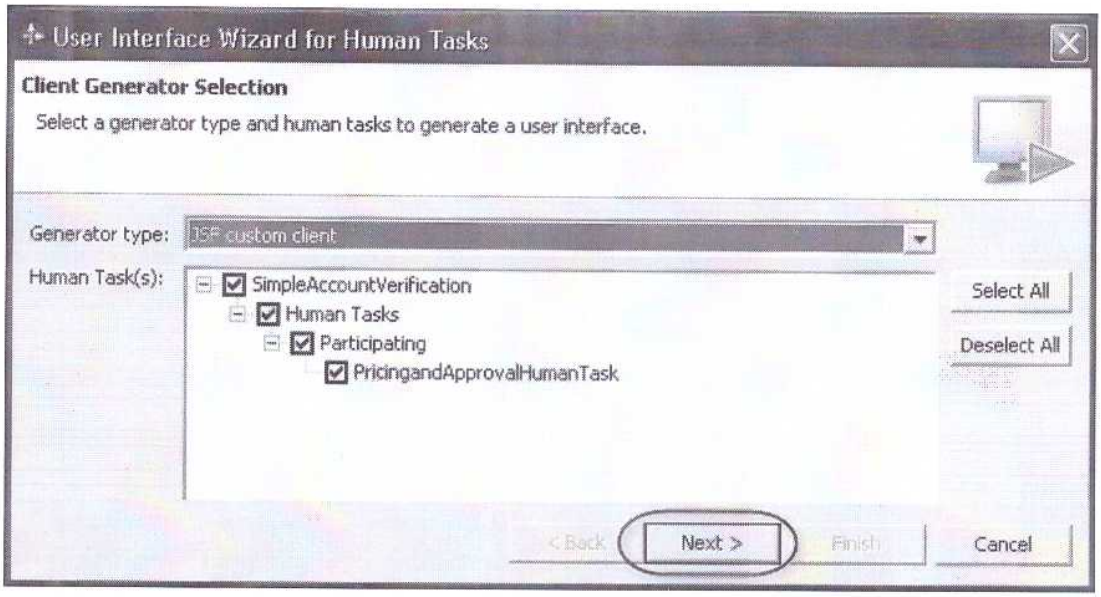
___k. Focus should return to the Assembly Diagram editor. Press **Ctrl-s** to save your work.

____ 3. Generate a Web User Interface for the Human Task

___a. From the **Business Integration** tree view, right-click on **SimpleAccountVerification**. From the popup menu, select **Generate User Interfaces**.

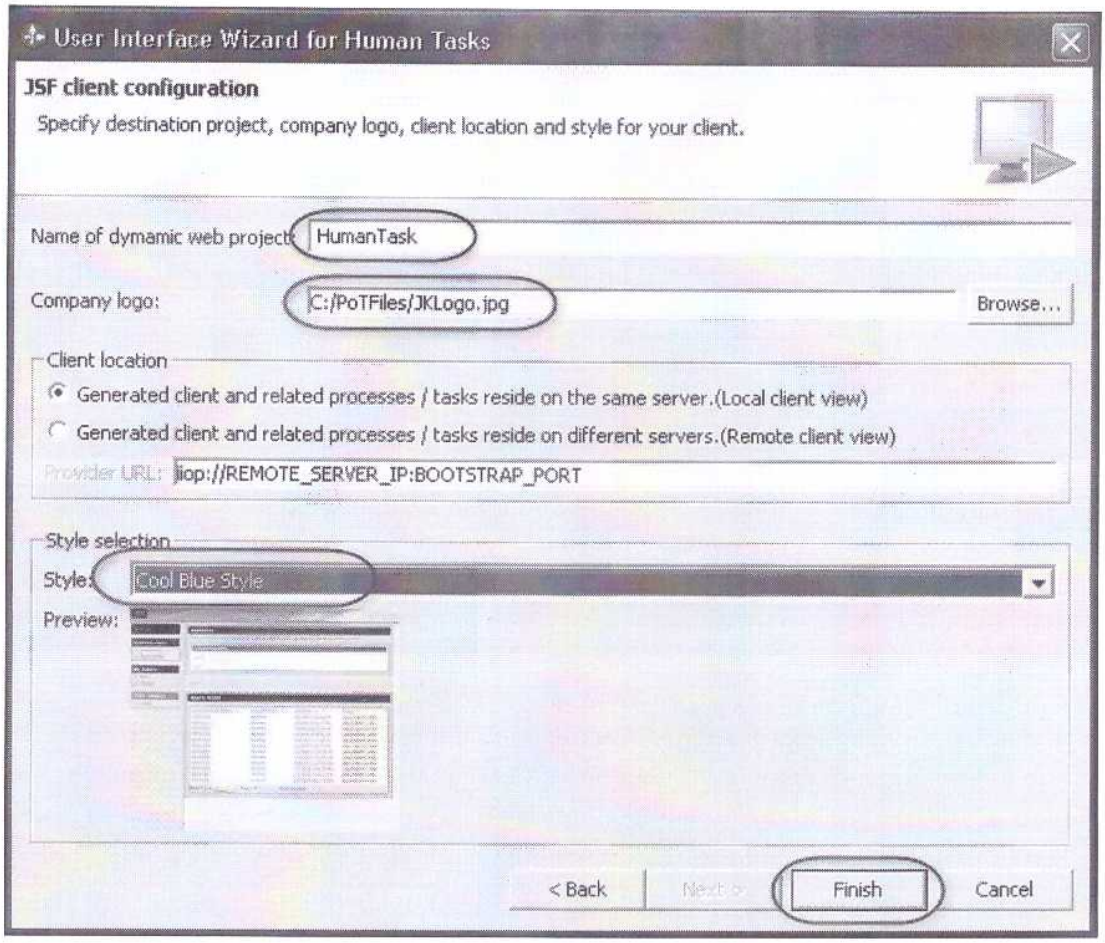


___b. From the **User Interface Wizard for Human Tasks** window, click on **Next**.



___c. From the next **User Interface Wizard for Human Tasks** window, specify the following, then click on **Finish**.

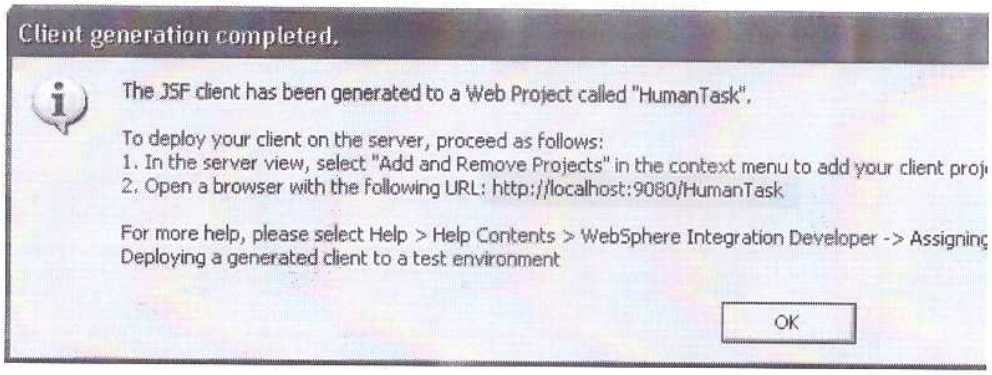
Name of dynamic web project:	**HumanTask**
Company logo:	**C:/PoTFiles/JKLogo.jpg** (Use the **Browse** button)
Style selection:	**Cool Blue Style**



This will generate a standard web page which can be accessed using a web browser. The web page and all its related artifacts will be packaged into its own J2EE project so that it can be deployed as a separate web application.

Let's pretend to work for a company called **JK Enterprises**. This is why we are using the graphic image file **JKLogo.jpg** for the Company logo.

___d. From the **Client generation completed** information window, click on **OK**.

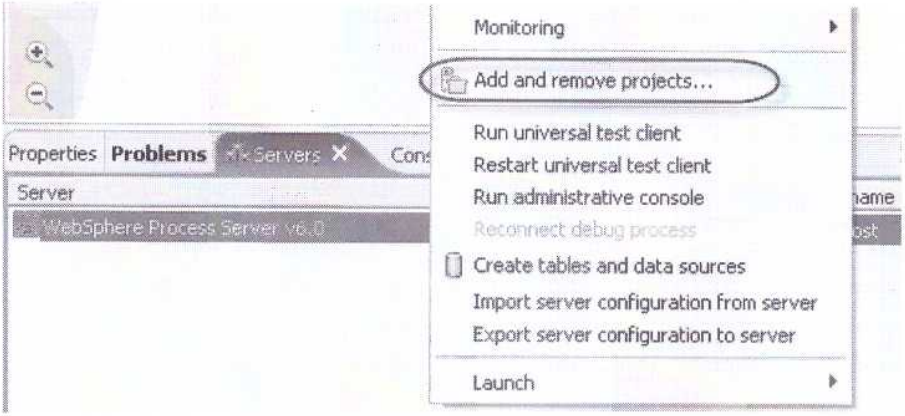


Take note of the URL mentioned in the information window (http://localhost:9080/HumanTask) which will access the Human Task web page. In this lab environment, the port number 9080 is actually 9083. The correct URL is http://localhost:9083/HumanTask.

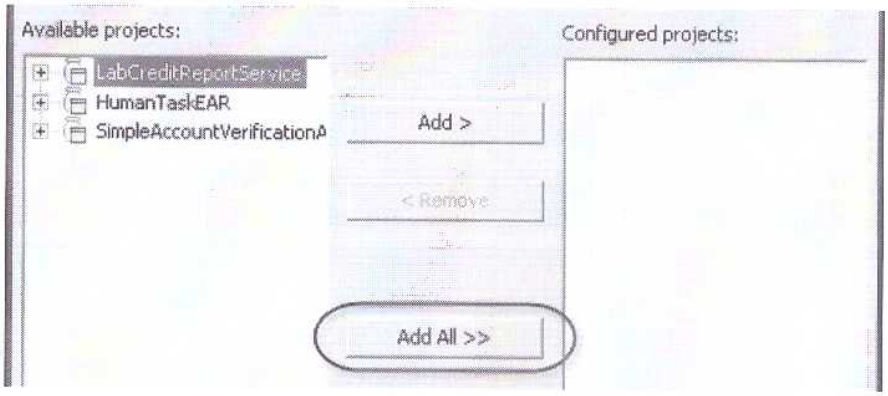
____ 4. Retest the SimpleAccountVerification process

___a. Switch to the **Servers** view.

___b. Right-click on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Add and remove projects**.



___c. From the **Add and Remove Projects** window, click on **Add All**.

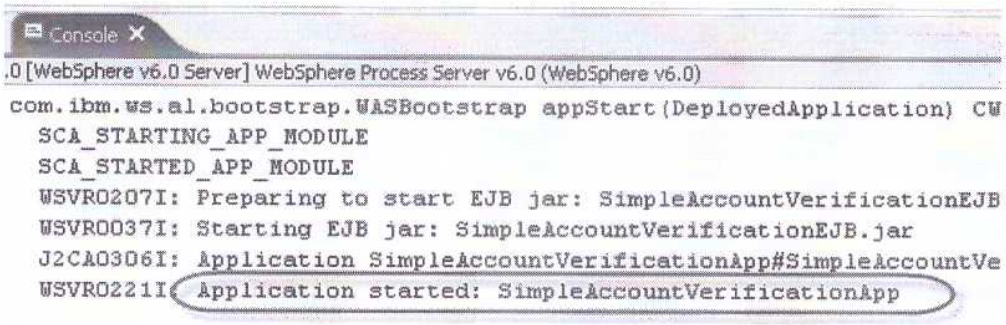


Take note of the new project **HumanTaskEAR**. This contains the generated Human Task web interface.

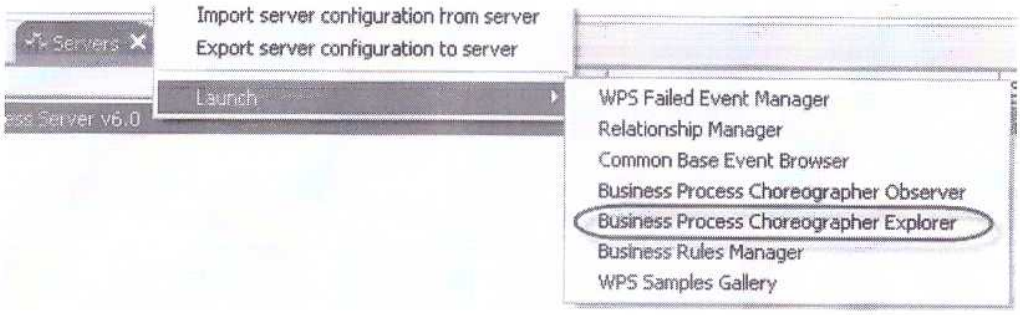
___d. Click on **Finish**.

___e. A progress indicator will appear in the lower-right corner of the WebSphere Integration Developer window. Wait for this indicator to disappear before proceeding to the next step. This will take a few minutes.

___f. Focus will automatically switch to the **Console** view. Verify that a message appears in the **Console** view indicating that the **SimpleAccountVerificationApp** process has started. Scroll right as needed.

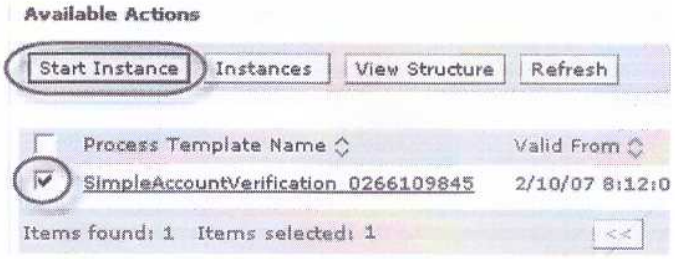


___g. Switch to the **Servers** view and right-click on **WebSphere Process Server v6.0**. From the popup menu, select **Launch->Business Process Choreographer Explorer**.



___h. Click on the **My Process Templates** link to display a list of processes which can be started.

___i. Place a checkmark beside the **SimpleAccountVerification** process template, then click on **Start Instance**.



___j. From the default input page, click on the Expand (+) button.

___k. For the **customerID** field, type **111**, accept the defaults for the other fields, then click on **Submit**.

Any Customer ID value other than "**123**" is required so that the request will be approved and follow the execution path which includes the **Pricing And Approval** business activity.

The Credit Score will be displayed, but the execution of the process will pause.

```
O **********************
O * Call Web Service  *
O **********************
O >>>>>>>>>>>>>>>>>>>>>
O >>>>> Credit Score = 501
```

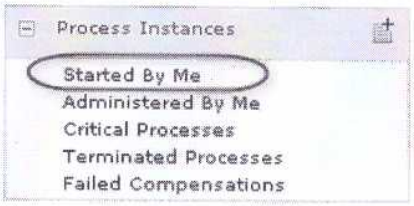
The process has actually reached the **Pricing and Approval** human task activity in the execution

flow. It is waiting for a response from a 'human'. In the next few steps, you will assume the role of a Loan Officer or Credit Manager, and interact with the business process.

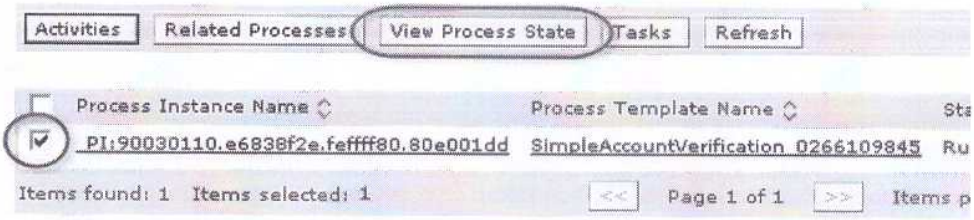
However, let's first verify that the process is actually waiting on the Human Task activity.

____ 5. View the Process State

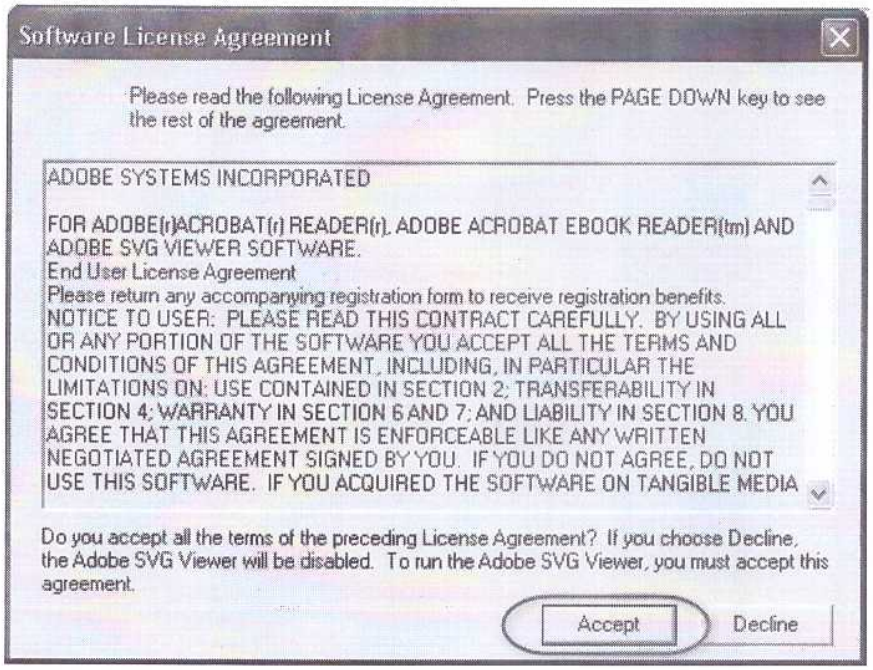
___a. From the **Business Process Choreographer Explorer**, click on the **Started By Me** link.



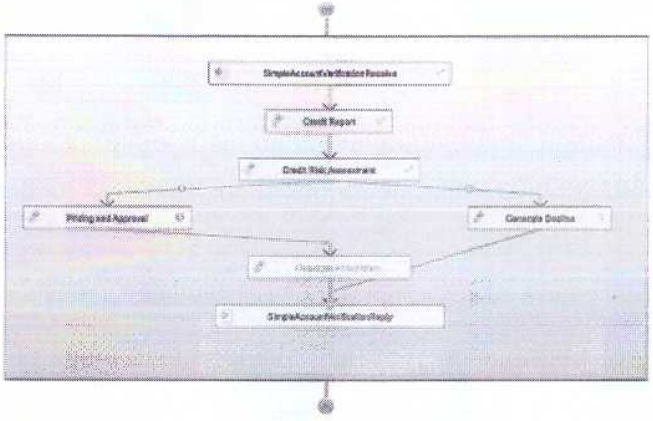
___b. In the **Available Actions** section, place a checkmark beside the process instance, then click on **View Process State**.



___c. Click on **Accept**.

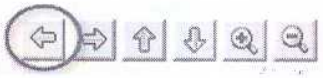


This will accept the license agreement to use the Adobe SVG Viewer. A graphic image of the current state of the SimpleAccountVerification process will be displayed in the SVG format.



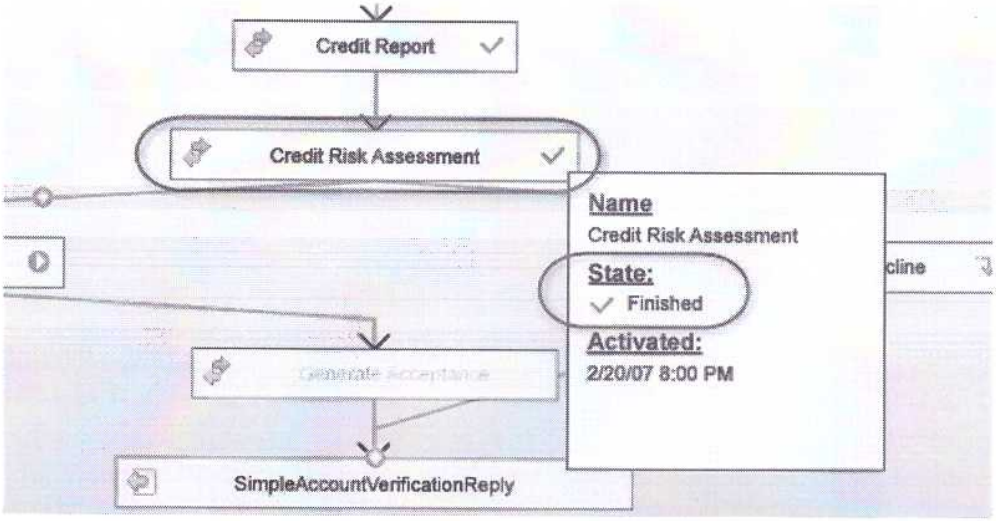
___d. Click on the **Zoom In** button above the SVG viewer two times.

___e. Click on the **Left** arrow button two times.



This should center the diagram in the SVG viewer. Continue to adjust and resize the image as necessary.

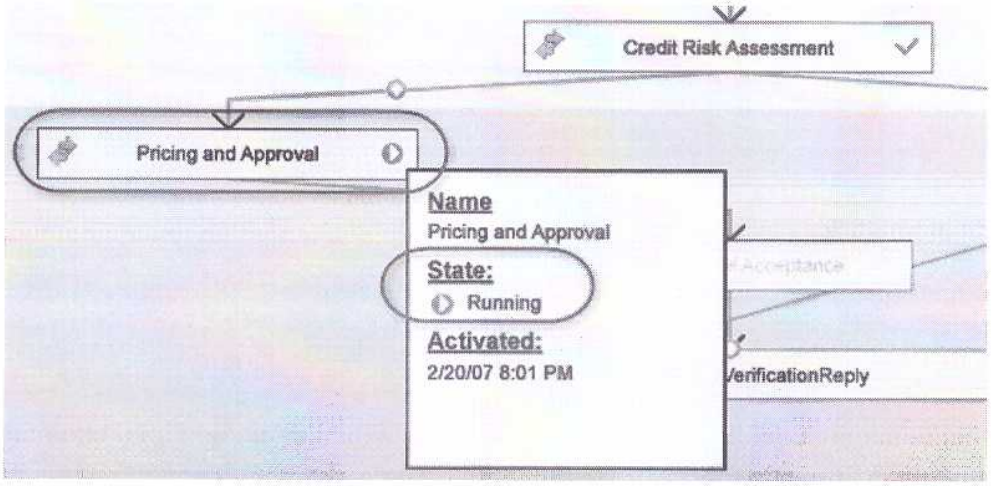
___f. Move the cursor on top of the **Credit Risk Assessment** task until a popup status box appears.



This shows that the **Credit Risk Assessment** task was invoked and completed.

___g. Move the cursor on top of the **Generate Decline** task. Note that the status box indicates that this activity was skipped because the request is going to be approved.

___h. Move the cursor on top of the **Pricing and Approval** task.



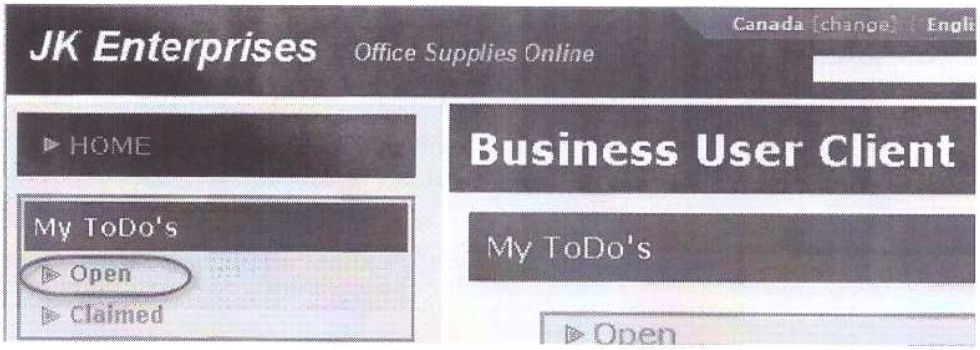
Note that the status is "**Running**", indicating that the process is currently waiting for this activity to finish. Since this is a **Human Task**, the process is waiting for a 'human' to respond. This will be done using the generated web interface in the next few steps.

____ 6. Use the Human Task Web User Interface

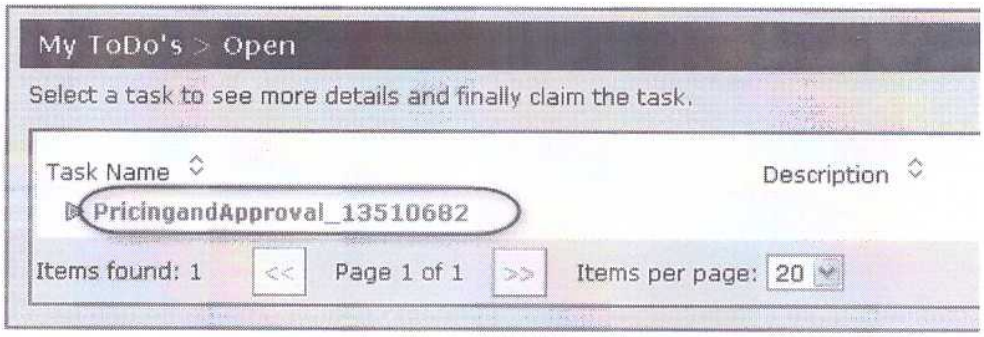
___a. Start the **Internet Explorer**.

___b. Go to **http://localhost:9083/HumanTask**.

___c. From the **My ToDo's** section, click on the **Open** link.

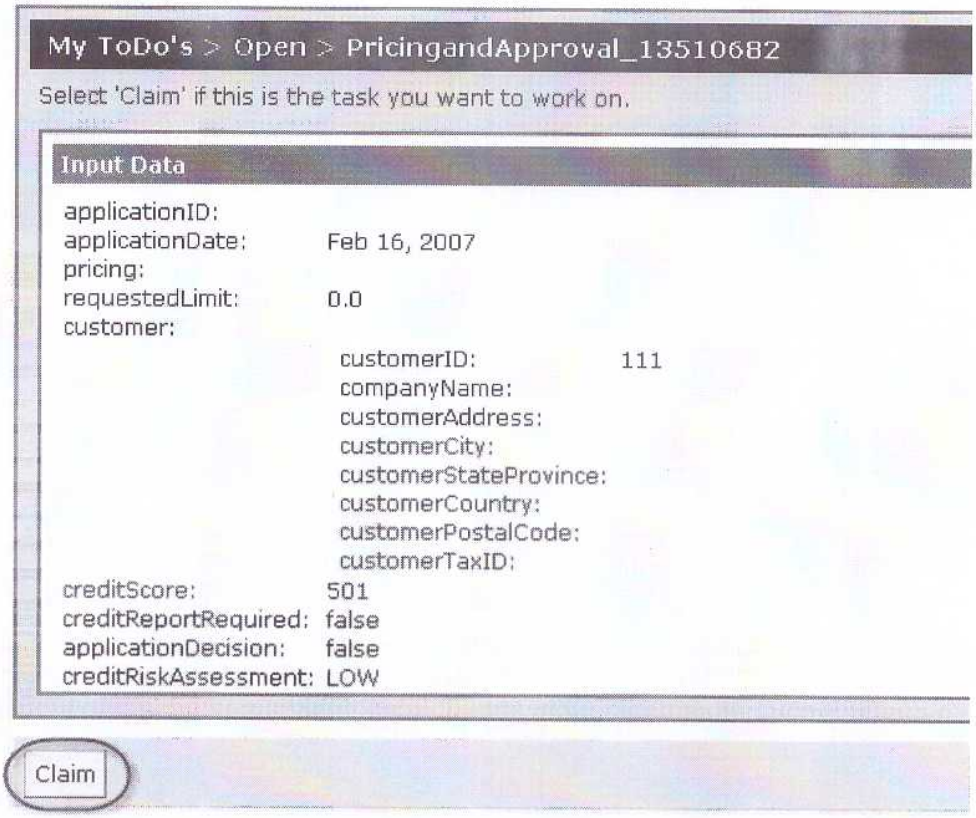


___d. From the **Open Tasks** section, click on the **PricingandApproval** task.

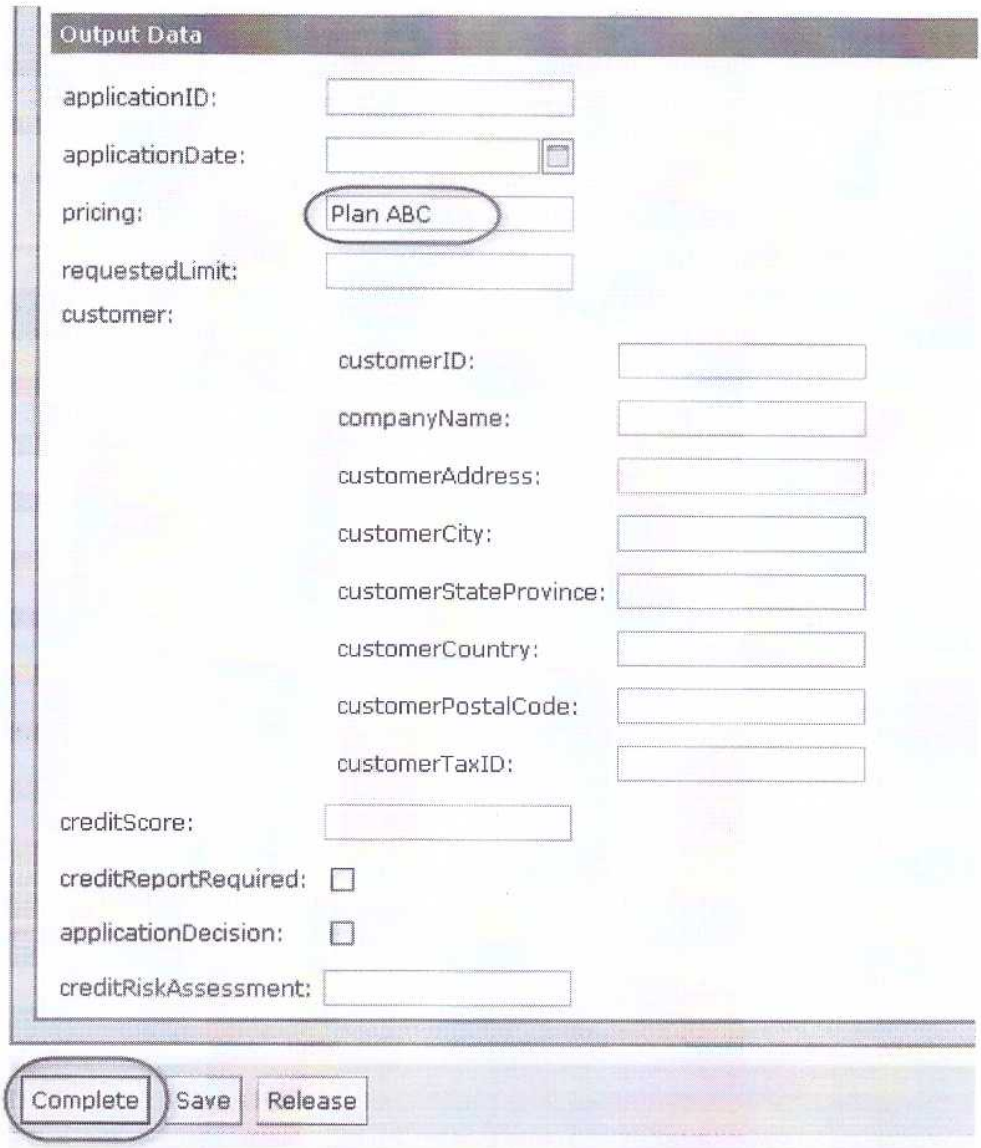


As previously mentioned, you were not required to log in with a UserID and password because security is currently disabled. This will typically not be the case in production environments. Based on the security profile or credentials of the user, the list should only display tasks which have been assigned to the user.

___e. Click on the **Claim** button.



___f. In the **Output Data** section, specify ***Plan ABC*** in the **pricing** field, then click on **Complete**.



You can actually specify any text you want in the **pricing** field. This will just be displayed as a message in the **Console** view. When the **Complete** button is pressed, the **Output Data** specified above will be passed back to the **SimpleAccountVerification** process, and the process will resume its execution.

___g. Switch back to the **WebSphere Integration Developer**.

___h. Switch to the **Console** view and check for the latest messages.

```
O >>>>>>>>>>>>>>>>>>>>>>>>>>>>
O >>>>> Request is approved!.
O >>>>> Pricing Plan has been set to Plan ABC
O >>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

The new messages will show the pricing plan specified in the **Pricing and Approval** human task.

____ 7. Cleanup

___a. Close the **Business Process Choreography Explorer**.

___b. Close the **Internet Explorer**.

___c. Switch to the **Servers** view.

___d. Right-click on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Add and remove projects**.

___e. From the **Add and Remove Projects** window, click on **Remove All**.

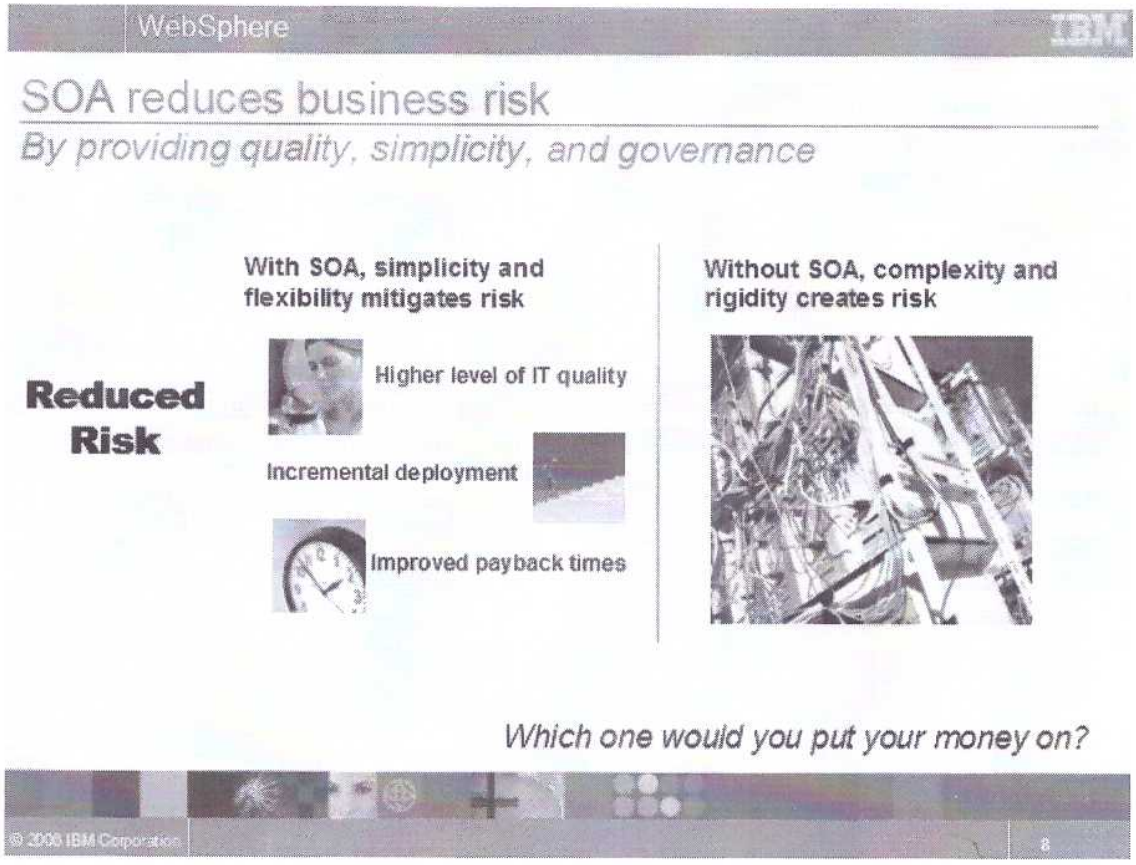
___f. Click on **Finish**.

___g. If a progress indicator appears, wait until it is complete.

___h. From the **Servers** view, right-click again on the **WebSphere Process Server v6.0** test server. From the popup menu, select **Restart -> Start**.

Please wait for the next lecture before proceeding with the lab.

Part 10: Conclusion



Hopefully these labs helped illustrate the significant benefits which can be achieved using IBM's SOA Foundation and the WebSphere Process Integration software portfolio.

- Business flexibility
- Better business processes
- Easier integration
- Reuse of assets
- Reduction of risk

Part 11: Appendix

1. Loading Solutions for Parts of the Lab

In case you encounter problems, or just wish to skip ahead to the solution of a specific part in the lab, then the completed projects are also available. The solutions can be imported as Project Interchange files located in the following directory:

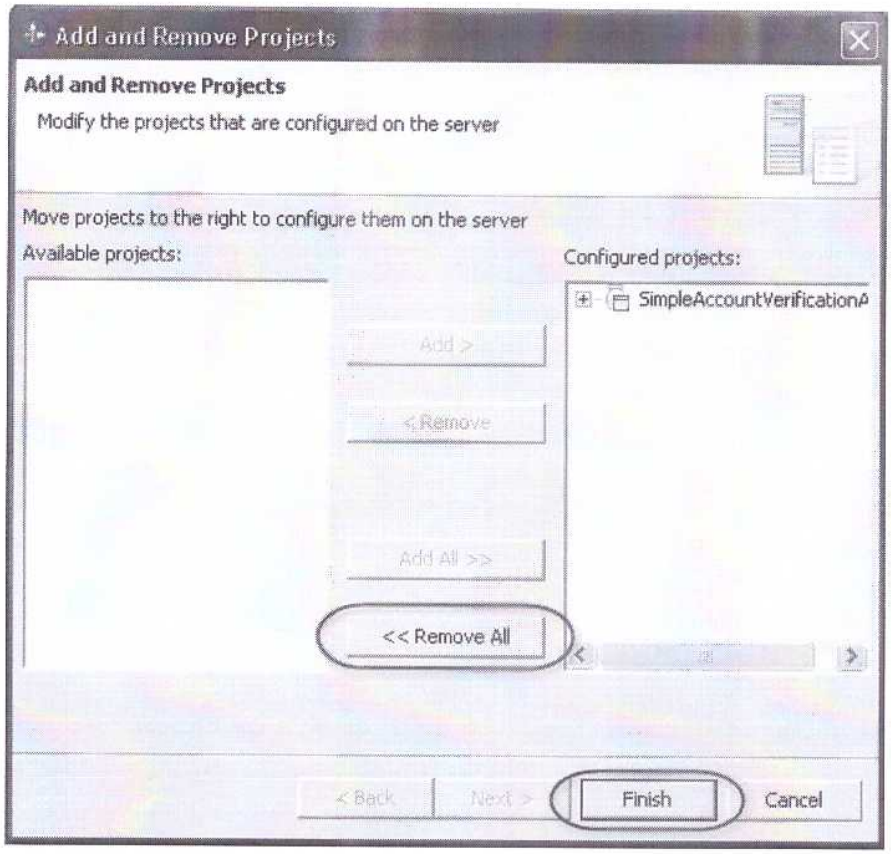
C:\PoTFiles\Solutions

Below is a list of the solution files which can be found in the directory above.

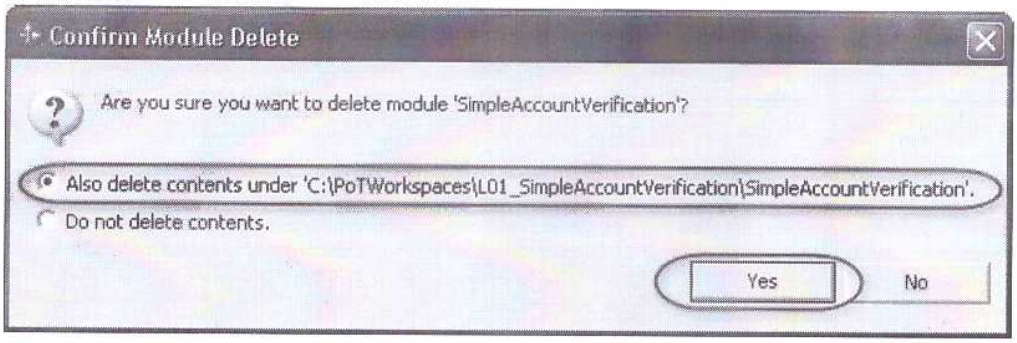
SimpleAccountVerificationSolution.mar	Part 2 fully completed (WebSphere Business Modeler)
SimpleAccountVerificationAllJava_PI.zip	Part 4 fully completed
SimpleAccountVerificationWebService_PI.zip	Part 5 fully completed
SimpleAccountVerificationBusinessRules_PI.zip	Part 6 fully completed
SimpleAccountVerificationHumanTask_PI.zip	Part 7 fully completed
SimpleAccountVerificationESB_PI.zip	Part 8 fully completed

Steps for loading a solution file into the WebSphere Integration Developer:

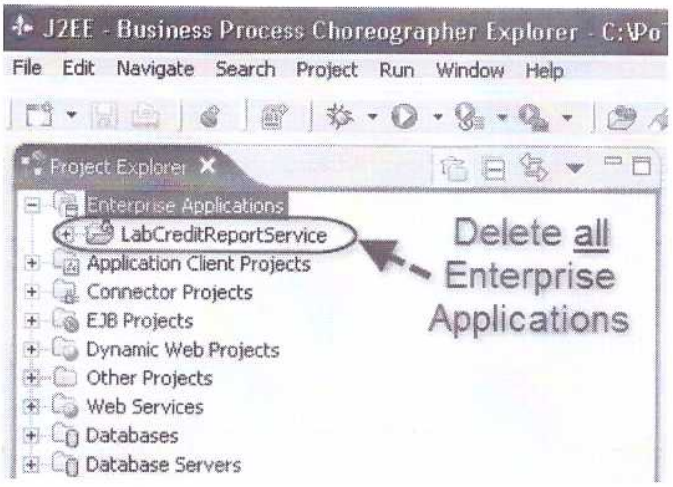
1. Ask for assistance from the instructors or lab assistants if necessary.
2. Remove all applications currently deployed to the test server. (Refer to the following screenshot)

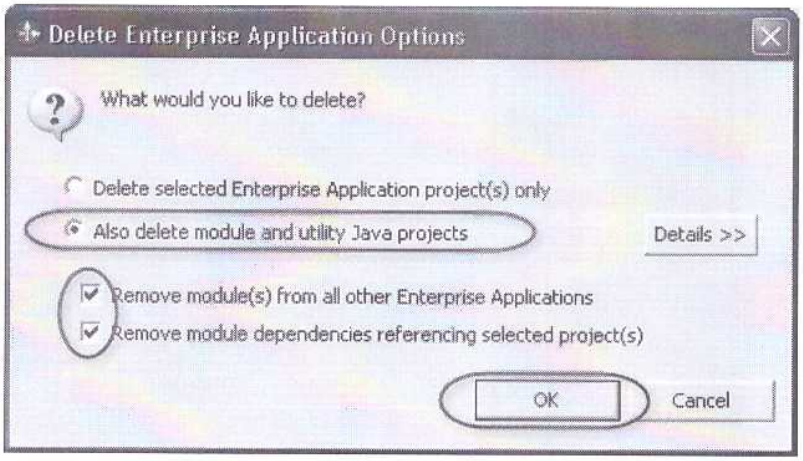


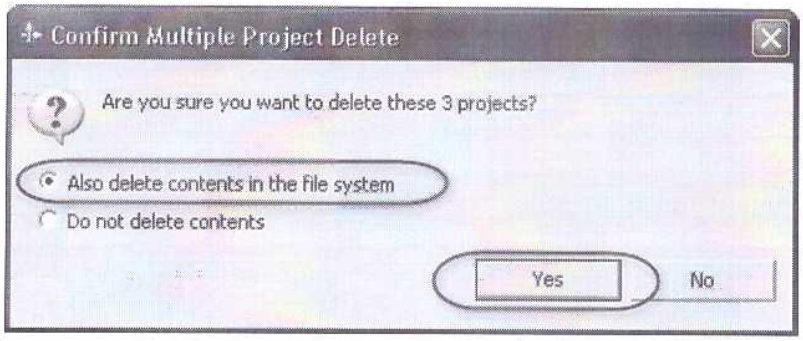
3. Delete **all** projects in the **Business Integration** view.



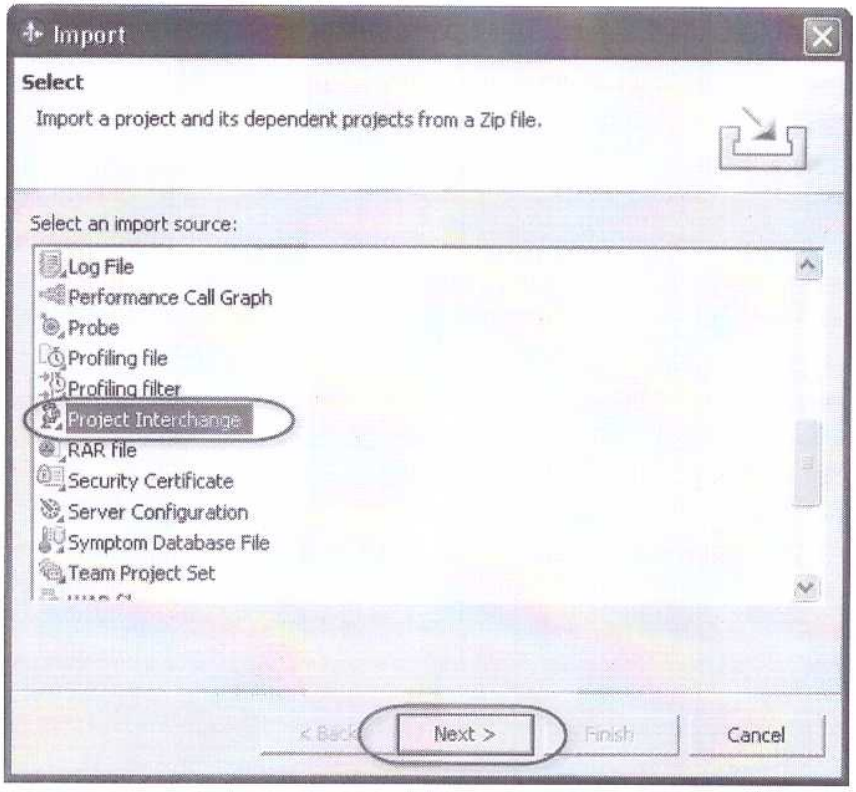
4. Switch to the **J2EE** perspective. Delete **all** enterprise applications.

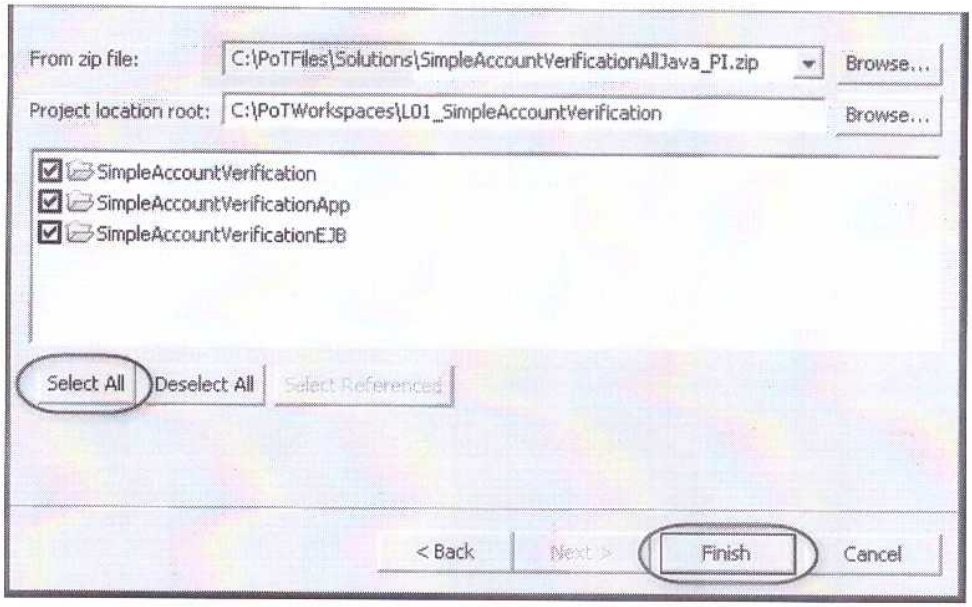




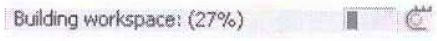


5. Switch back to the **Business Integration** perspective.
6. Import the desired Project Interchange file. Refer to the solution file table described above.

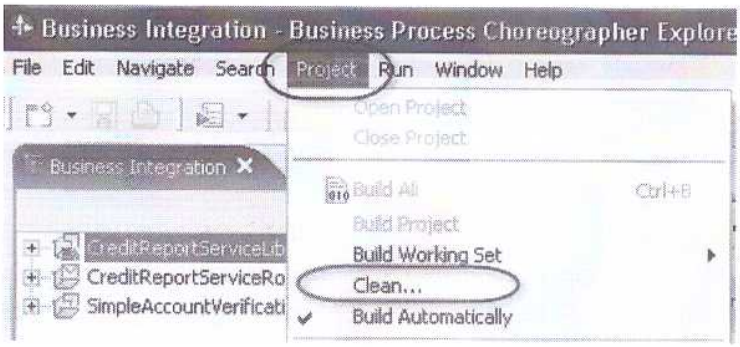


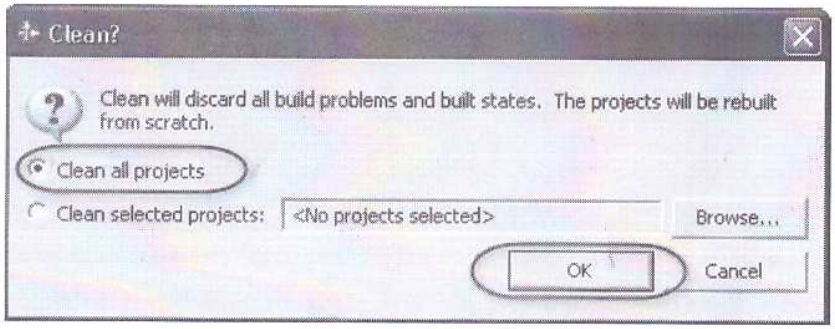


7. If a progress indicator appears, wait until it is complete.

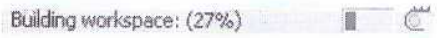


8. Perform a Project Clean-all.





9. If a progress indicator appears, wait until it is complete.



10. Ensure that there are no errors listed in the Problems view. If errors exist, delete the following files and perform another Project Clean-all (repeat step 8):

    SimpleAccountVerificationApp
    SimpleAccountVerificationEJB
    SimpleAccountVerificationWeb

    Do not delete SimpleAccountVerification.

11. Start the test server if not yet started.
12. Add all projects to the test server.
13. Proceed to the section for testing the process in the appropriate part of the lab.